USING UPPER LAYER WEIGHTS TO EFFICIENTLY CONSTRUCT AND TRAIN FEEDFORWARD NEURAL NETWORKS EXECUTING BACKPROPAGATION

THESIS

Harmon J.A. Gage, Second Lieutenant, USAF

AFIT-OR-MS-ENS-11-06

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-OR-MS-ENS-11-06

USING UPPER LAYER WEIGHTS TO EFFICIENTLY CONSTRUCT AND TRAIN FEEDFORWARD NEURAL NETWORKS EXECUTING BACKPROPAGATION

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Harmon J.A. Gage, BS

Second Lieutenant, USAF

March 2011

AFIT-OR-MS-ENS-11-06

USING UPPER LAYER WEIGHTS TO EFFICIENTLY CONSTRUCT AND TRAIN FEEDFORWARD NEURAL NETWORKS EXECUTING BACKPROPAGATION

Harmon J. A. Gage, BS
Second Lieutenant, USAF

Approved:

__//Signed____________________________	16 MAR 2011 ___
Dr. Kenneth. W. Bauer (Chairman)	date
__//Signed____________________________	16 MAR 2011 ___
Dr. John Miller (Reader)	date

AFIT-OR-MS-ENS-11-06

Abstract

Feed-forward neural networks (FFNN) executing back propagation are a common tool for regression and pattern recognition problems. These types of neural networks can adjust themselves to data without any prior knowledge of the input data. FFNNs with a hidden layer can approximate any function with arbitrary accuracy.

In this research, the upper layer weights of neural networks are used to determine an effective middle layer structure and when to terminate training. By combining these two techniques with signal-to-noise ratio feature selection, a process is created to construct an efficient neural network structure. The results of this research show that for data sets tested thus far, these methods yield efficient neural network structure in minimal training time. Data sets used include an XOR data set, Fisher's iris data set, and a financial industry data set, among others.

AFIT-OR-MS-ENS-11-06

Dedicated to all those who have been a part of my education

Acknowledgments

There are many people I would like to acknowledge for allowing me to successfully complete this research. First, I would like to express my gratitude to my advisor, Dr. Kenneth Bauer, for guiding me through the research process and sharing some of his vast knowledge with me along the way. Without him, this effort would not have come together. I would also like to thank my reader, Dr. John Miller, for his feedback and support.

Next, I would like to give a special thanks to Capt Jason Williams for being an honorary member of my committee and for answering all my questions during the research process. He and Capt Ben Hartlage taught me that sometimes the answers to life's most difficult questions are efficiently stacked inside the Joe Price memorial fridge.

Further, I thank my family and my previous educators who have brought me to this point in my academic career. My parents always pushed me to pursue further education and provided me reassurance when I doubted what I was doing along the way. For this reason, I believe the accomplishment of this thesis was not started at AFIT, but at a very young age.

Also, I would like to thank Lt Col Brian Ralston, my USAFA academic advisor, for pushing me to pursue my Master's degree as soon as possible.

There are surely others who I have failed to mention, but know that I am grateful to everyone who has been a positive influence on my life throughout this process. You make it all worth it.

-Harmon J.A. Gage

Table of Contents

Page

List of Figures

List of Tables

USING UPPER LAYER WEIGHTS TO EFFICIENTLY CONSTRUCT AND TRAIN FEEDFORWARD NEURAL NETWORKS EXECUTING BACKPROPAGATION

1. Introduction

1.1. Background

In general, the size of a neural network directly affects both network complexity and learning time. The size of the network also impacts the ANN's ability to operate effectively with data not contained in the training set (Bebis & Georgiopoulos, 1994). Bebis & Georgiopoulos compare the process of determining the correct number of hidden nodes to curve fitting a set of data points. If the curve is a high-order polynomial and intersects every point in the data set, the curve likely will not be accurate when introduced to new data. On the other hand, if a curve does not fit the training set with moderate success it likely will not be accurate when introduced to new data either. Finding a balance between these two scenarios is, for lack of a better term, an art. Determining the number of hidden nodes acts in a similar fashion. Too many neurons and the network will over-learn data, also known as over-training, but too few neurons will not provide an ideal model either (Kavzoglu, 1999). Many have researched the problem of determining the "best" number of hidden neurons. The different approaches can be split into five categories: constructive algorithms, pruning algorithms, trial and error, empirical formulas, and hybrid methods. Constructive algorithms start training a network with a small number of middle layer nodes and add structure as necessary. Pruning algorithms start training a network with an overly large number of middle layer nodes and prune them off. Trial-and-error requires no structured thought to how or why a given number of hidden neurons are used. Empirical formulas are self explanatory in that they determine the number of hidden nodes with a formula. These formulas usually incorporate the data set of interest's number of variables and classes. Hybrid techniques use a combination of constructive and pruning algorithms along with other methods such as genetic algorithms and singular value decomposition (Ileana, Rotar, & Incze, 2004) (Teoh, C, & Xiang, 2006).

Selecting the correct number of hidden layer nodes for a neural network is not the only work concerning the structure of neural networks that has been conducted. Several researchers have investigated ways of selecting which inputs are important to the classification process.   This is known as feature selection. An approach introduced by Bauer, Alsing, & Greene, uses signal-to-noise ratios (SNR) to accomplish feature selection (Bauer, Alsing, & Greene, 2000). The approach using SNR is the motivation behind the research in this paper. The SNR method, discussed in more detail in section 2.7.3, gathers data from the lower layer weights. This information is used to determine which features are actually important. The work in this research is an analysis of the upper layer weights, which connect the middle layer neurons to the outputs.

The upper layer weights of a network can also be analyzed to gain an understanding of when a network is sufficiently trained. When to terminate training is an open problem. If a network is trained for too few epochs, the network is not able to succesfully classify data. If a network is trained for too many epochs, time is wasted during the training process. In this research, the upper layer weights of a network are evaluated to gain insight regarding the progress of training.

Combining the insight upper layer weights provide regarding structure size and training status of a neural network with SNR feature selection provides the building blocks of a process to train a succesful, efficient neural network structure.

1.2. Problem Statement

There are many methods for determining the best number of hidden nodes for a multi-layer neural network. Currently, however, there has not been much effort to analyze upper layer weights to accomplish this during the training process of a network. This research will introduce an approach to determine an efficient number of hidden nodes in a multi-layer feed forward neural network. The upper layer weights should provide information regarding the quality of neural networks being used and the progress of the training process. Selecting a middle layer structure, terminating training, and removing non-salient features (via the SNR method) are all necessary to build a successful network structure. A successful neural network not only learns the data it is trained with, but can also generalize well. An iterative, structured process to construct a neural network that generalizes well would be a useful tool to minimize time experimenting with neural network attributes.

1.3.Research Objectives

The focus of this effort is to construct quality neural networks using upper layer weights. In particular, the correct middle layer structure is of interest. After arriving at a method of determining an "efficient" number of hidden layer nodes, the method will be compared to a current constructive algorithm and empirical formulas. The objective is to provide a method regarding training that is efficient and avoids over learning the data.

The culmination of this work will be to combine these methods with SNR feature selection to build a structured process to follow in selecting an efficient, successful neural network in a timely manner. A fusion technique will also be introduced to yield the best possible results.

2. Literature Review

This chapter contains definitions of common neural network terms, an explanation of multi-layer perceptrons, back propagation, discussion of current methods to determine neural network structure, pruning/constructive algorithms, and a discussion of feature selection techniques. The feature selection techniques are included, because these techniques can be adapted to hidden layer structure selection as well.

2.1. Definitions

Back-propagation method: A learning algorithm for updating weights in feed forward neural network that minimizes mean squared mapping error. (Svozil, Kvasnicka, & Jiri, 1997)

Constructive Algorithms: Process that begins with a small number of hidden neurons, trains the network, before adding hidden neurons as necessary prior to retraining.

Epoch: A complete set of data used in the training of a neural network after one full cycle; also known as a training cycle. (Steppe J. M., 1994)

Exemplar: One observation of data input into a neural network. Contain one value for each feature.

Feature: The individual entries found in exemplars. They contain information that can prove helpful in classifying exemplars. In other fields, features are known as attributes and independent variables. (Steppe J. M., 1994)

Feed-Forward: A network in which each node in a given layer is forward connected to every node in the next layer. (Steppe J. M., 1994)

Hidden layer: A layer of neurons located between the input layer and the output layer.

Hidden neurons: In a multi-layer network these are the neurons that are present in the "hidden layers". In other words, any neuron that is not an input neuron or an output neuron is a hidden neuron. (Steppe J. M., 1994)

Hybrid techniques: A combination of constructive & pruning algorithms along with other methods (such as genetic algorithms and singular value decomposition) used to determine the best number of hidden layer neurons. (Steppe J. M., 1994)

Multilayer Perceptron: A multilayer feed-forward network that is fully connected. Any neuron in a given layer is connected to every neuron in the next layer. They are the most widely used and studied neural network classifiers. (Zhang, Jiang, Liu, Liang, & Yu, 1997)

Neural Network: An important statistical tool for classification. They can adjust themselves to data without any prior knowledge of the input data. They can approximate any function with arbitrary accuracy. (Zhang G. P., 2000)

Pruning Algorithm: Process that starts training with a larger than necessary number of hidden neurons, remove the neurons that are not needed, and retrain until success is obtained. (Reed, 1993)

2.2. Feed-Forward Neural Network

Feed-forward neural networks are generally used in two types of applications. They are used to estimate a linear or nonlinear function for prediction in regression analysis and to estimate a linear or nonlinear discriminant function for classification in discriminant analysis (Steppe J. M., 1994)

As stated in the definitions, a multi-layer perceptron (MLP) is a multilayer feed-forward network that is fully connected. This research will specifically involve multi-layer perceptrons.

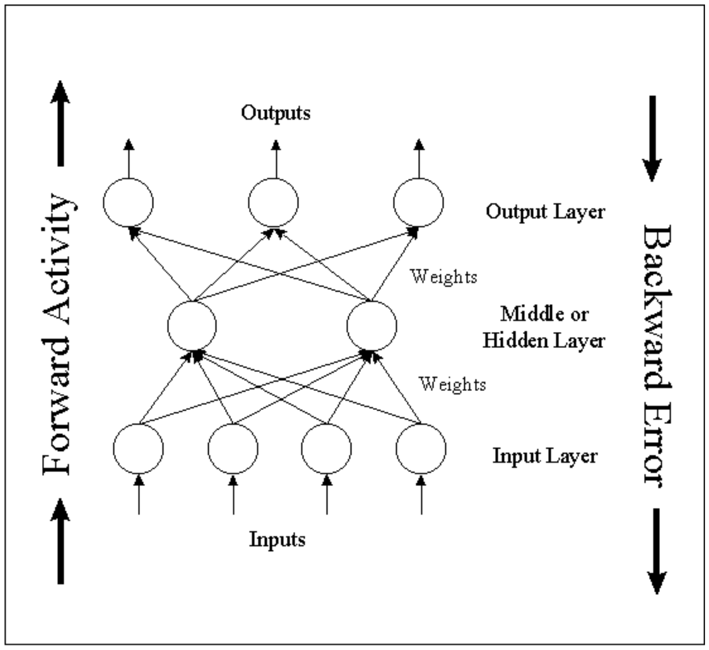


Figure 2-1 Single Hidden Layer Feedforward Neural Network (Caudill & Butler, 1992)

Figure 2-1 displays that neurons residing in different layers are connected by weighted synapses. The input layer contains a number of neurons that corresponds to the number of inputs for any/all exemplars in the data set. Normalized feature input data is passed through the input layer. This data can be the raw data or some transformation/projection of it.

The hidden layer nodes have a linear or nonlinear activation function. Linear functions are generally used for regression analysis and nonlinear for discriminant analysis. The functions receive weighted variations of the input data. The specific activation function used in this research is discussed in section 2.3. After passing through the activation function, the output again “travels” along weighted synapses toward the output nodes.

The output layer nodes also have linear or nonlinear activation functions (again depending on the application). Just like the hidden layer, linear activation functions are generally used for function approximation and nonlinear functions for discriminant function applications (Steppe J. M., 1994).

2.3. Backpropagation

As stated previously, multi-layer perceptrons (MLPs) are the most popular neural networks. MLPs all execute the backpropagation algorithm. Lippmann explains the backpropagation training algorithm as “an iterative gradient algorithm designed to minimize the mean square error between the actual output of a multilayer feed-forward perceptron and the desired output” (Lippmann, 1987). The actual instantaneous backpropagation algorithm is available below (Steppe J. M., 1994). This is an instantaneous algorithm, because the weights are updated after each exemplar is introduced to the network. Another version of backpropagation waits until an entire epoch is complete before calculating the error gradient.

**The Instantaneous Backpropagation Algorithm
for a
Single Hidden Layer Feedforward Neural Network.**

1. Randomly partition data into *training, training-test,* and *validation* sets.
2. Normalize the feature input data.
3. Initialize weights to small random values.
4. Present the network with a randomly selected vector from the training set, denoted $\mathbf{x}^\mathbf{p}$
5. Calculate the network output $\mathbf{z}^\mathbf{p}$ associated with the *p*th training vector

    - *k*th neural network output: $z_k^p = f(\sum_{j=0}^{H} w_{jk}^2 x_j^1)$, where

    - H is the number of middle nodes

- $f(\alpha) = 1/(1+e^{-\alpha})$ for sigmoidal activation function
- $f(\alpha) = \alpha$ for linear activation functions
- w^2_{jk} is the weight from middle node *j* to output node *k*
- x^1_0 is the middle layer bias term and is set equal to 1
- $x^1_j = f(\sum_{i=0}^{M} w^1_{ij} x^p_i)$ is the output of middle node *j*
- M is the number of feature inputs (input neurons)
- w^1_{ij} is the weight from input node *i* to middle node *j*
- x^p_0 is the input layer bias term, and is equal to 1
- x^p_i is the *i*th feature input

6. Update the weights
    - Output layer weights: $(w^2_{jk})^+ = (w^2_{jk})^- + \eta\delta^2_k x^1_j$,
    - Input layer weights: $(w^1_{ij})^+ = (w^1_{ij})^- + \eta\delta^2_j x^p_i$,where
    - $(w^2_{jk})^+$ is the updated weight from middle node *j* to output *k*
    - $(w^2_{jk})^-$ is the old weight from the middle node *j* to output *k*
    - $(w^1_{ij})^+$ is the updated weight from input *i* to middle node *j*
    - $(w^1_{ij})^-$ is the old weight from input *i* to middle node *j*
    - η is the step size
    - $\delta^2_k = (d^p_k - z^p_k) z^p_k (1 - z^p_k)$ if there is a sigmoid on the output
    - $\delta^2_k = (d^p_k - z^p_k)$ if the output is linear
    - $\delta^1_j = x^1_j (1 - x^1_j) \sum_{k=1}^{K} \delta^2_k (w^2_{jk})^-$ if there is a sigmoid on the middle node *j*
    - $\delta^1_j = \sum_{k=1}^{K} \delta^2_k (w^2_{jk})^-$, if middle node *j* is linear
    - d^p_k is the *k*th desired output of the *p*th exemplar
7. If training-test set error does not indicate sufficient convergence, go to step 4.

This algorithm is called the back-propagation algorithm, because the output error propagates from the output layer through the hidden layers to the input later (Svozil, Kvasnicka, & Jiri, 1997).

Step 1 randomly divides the problem data into a *training* data set and a *validation* set. *Training* data is used to create weight parameters within the model. The validation data is used to evaluate the neural network's ability to generalize (work accurately on data it has not seen before).

Step 2 normalizes the data in both the validation and training sets.

Step 3 initializes weight parameters randomly between -.1 and .1. This is the interval used in this research, but it is by no means a concrete rule.

Step 4 introduces a randomly selected vector $\mathbf{x}^{\mathbf{p}}$ to the neural network. *p* designates the *p*th vector *x* in the training set.

Step 5 calculates the vector of network outputs. The sigmoidal activation function $f(\alpha) = 1/(1+e^{-\alpha})$, known as a squashing function, is used because its derivative is continuous and makes the weight update rule simple for the backpropagation training (Steppe J. M., 1994). Also, previous work by (Cybenko, 1988) shows that a multi-layer neural network executing backpropagation with linear output nodes is capable of arbitrarily accurate approximations for any arbitrary function given a sufficient amount of hidden nodes are used with a sigmoidal activation function. The topic of "sufficient amount of hidden nodes" is a focus of this research.

Step 6 calculates the instantaneous output error of the neural network. The step size, η, can be a constant or a variable. White proposes that a constant learning rate is not efficient since randomness in the input, such as noise, causes fluctuations in the weight vector. This prevents backpropagation from converging to an optimal weight vector (White, 1993).

2.4.Constructive and Pruning Algorithms

Constructive algorithms and pruning algorithms are closely related. A constructive algorithm begins with a small number of hidden neurons and trains the network. Following a training cycle, a neuron is added to the hidden layer and the network is trained again. This process is repeated until performance is satisfactory (Frean, 1990). Pruning algorithms are essentially the opposite of constructive algorithms. Their approach is to train a network with a larger than necessary number of hidden neurons, remove the neurons that are not needed, and retrain until success is obtained (Reed, 1993). Success is defined by the user. As mentioned earlier there is a trade-off between minimizing error in the neural network and maintaining the network's ability to be a successful general model.

Steppe, Bauer, and Rodgers introduced a pruning type algorithm for hidden neuron selection using the likelihood ratio test statistic (Steppe, Bauer, & Rogers, 1996). This same algorithm is utilized by Belue, Steppe, & Bauer and Kocur, et al. (Belue, Steppe, & Bauer, April 1996) (Kocur, et al., 1996). Bacauskiene and Verikas introduce a feature selection procedure which performs a pruning type algorithm using multiple neural networks and eventually committees (multiple sets of neural networks used for classification) (Verikas & Bacauskiene, 2002) (Bacauskine & Verikas, 2004).
Two succesful constructive algorithms include *Upstart* algorithm and the *Cascade Correlation* algorithm.

The *Upstart* algorithm adds middle layer nodes if the current middle layer structure causes the network to get stuck in a local minimum when trying to minimize error. If the current structure gets "stuck", two more nodes are added (Frean, 1990). This algorithm is very succesful when attempting to solve binary problems as it is guaranteed to eventually classify each point correctly. However, this algorithm can lead to overlearning of the data. Fanguy & Kubat introduce modifications to the *Upstart* algorithm that allow it to solve multiclass problems (Fanguy & Kubat, 2002).

The *Cascade Correlation* algorithm also constructs a network structure without the use of backpropagation or error signals. Hidden units are added to the network one by one. Once training epochs no longer create a significant reduction in error, a new hidden neuron is added. Next, the input weights are altered to correlate the output of the hidden nodes with the network's output error. Once this is done, the whole network is retrained and the process is repeated until the total error is below a certain threshold (Fahlman & Lebiere, 1990). This process has proven succesful for real valued mappings (Bebis & Georgiopoulos, 1994). Yang & Honavar publish a paper with *Cascade Correlation* experiments including performance for Fisher's Iris data set. Results for the iris data from this work are compared to the *Cascade Correlation* results in 4.5.3.

The constructive algorithm proposed in this research follows similar logic to the *Upstart* and *Cascade Correlation* algorithms. However, instead of observing the error during training, the upper layer weights are observed to determine when enough structure is present.

2.4.1. Upper bound on hidden nodes for pruning algorithms

Huang and Babri proved that the number of hidden nodes for a neural network with a non-linear activation function should never be larger than the number of input samples (exemplars) (Huang & Babri, 1998). However, if the number of exemplars is excessive, the computational complexity of determining the correct sized ANN increases.

Other work provides a tighter bound on the number of hidden nodes provided some conditions are met. Cover introduces a rule based on separating capacities of families of nonlinear decision surfaces (Cover, 1965). He shows that a family of surfaces having *s* degrees of freedom has a natural separating capacity for 2*s* training exemplars. Unless there a more than 2*s* exemplars, vague generalization is likely. Cover's theorem gives the following upper bound on the number of middle nodes:

$$H < \frac{.5P - 1}{M + 1} \text{ (Cover, 1965)}$$

where P is the number of exemplars and M is the number of features. If the number of features is large and the number of exemplars is small, Cover's theorem could suggest fewer middle nodes than are required for the complexity of the problem. Cover's theorem's upper bound is often overly conservative and can call for more middle nodes than necessary. For this research, the middle layer structures tested are not bigger than Cover's criterion. We use Cover's theorem because it gives the largest cardinality of a set of training vectors such that any possible class assignment of the vectors can be implemented with probability one (Steppe, Bauer, & Rogers, 1996). Vapnkik and Cherconenkis (Vapnik & Chervonenkis, 1971), Baum and Haussler (Baum & Haussler, 1989), and Sontag (Sontag, 1992) provide other heuristics.

2.4.2. Lower bound on hidden nodes for constructive algorithms

The lower bound on hidden nodes for a constructive algorithm is normally one neuron (Sartori & Antsaklis, 1991) (Parekh, Yang, & Honavar, 2000). There are cases when 1 middle neuron is not sufficient in separating data, but this structure size is tested on each data set in this research.

2.5. Empirical Formulas

There are a few empirical formulas that are suggested as good heuristics to determine the “best” number of hidden nodes for a given data set. These formulas include, but are not limited to, the formulas listed in this section.

The following formulas refer to different neurons of the network structure as:

Hidden layer neurons: M

Input layer neurons: K

Outer layer neurons: J

Formula 1: According to the Kolmogorov theorem

$$M = 2*K+1 \text{ (Kurkova, 1992)}$$

Formula 2: Lippmann believes the number of hidden nodes should be

$$M = \sqrt{J*K} \text{ (Lippmann, 1987)}$$

when the number of inputs is larger than the outputs.

Formula 3: Zhang has determined the range of M as

$$M = \sqrt{J*K}+c \text{ (Zhang, Jiang, Liu, Liang, \& Yu, 1997)}$$

Where c is some constant between 1 and 10

Formula 4: Daqi and Shouyi determined the best number of hidden neurons is

$$M = \sqrt{K*(J+2)}+1 \text{ (Daqi \& Shouyi, 1998)}$$

Formula 5:

$$\sum_{i=0}^{J} \binom{M}{i} > P$$

$$\text{If } i\text{>M}, \binom{M}{i} = 0$$

Where P is the number of patterns or separable regions in the data (Mirchandani & Cao, 1989)

Formula 6: Gormon determined the best number of hidden neurons as

$$M = \log_2 P \text{ (Gorman \& Sejnowski, 1988)}$$

Gao, Chen and Qin introduced a method of using each of these empirical formulas and to create a lower and upper bound that contains the best possible middle layer size (Gao, Chen, & Qin, 2010).

2.6. Terminating Training

When to terminate training is an open problem. Since the algorithm used in feed-forward neural nets attempts to minimize mean square error, many training termination heuristics involve the mean square error of the validation set. The validation error is said to mimic the generalization error of a network. Prechelt provides a simple algorithm to terminate training:

1. Split the training data into a training set and a validation set, e.g. in a 2-to-1 proportion.
2. Train only on the training set and evaluate the per-example error on the validation set once in a while, e.g. after every fifth epoch.
3. Stop training as soon as the error on the validation set is higher than it was the last time it was checked.
4. Use the weights the network had in that previous step as the result of the training run.

Prechelt continues on and discusses that this type of algorithm can result in premature termination.

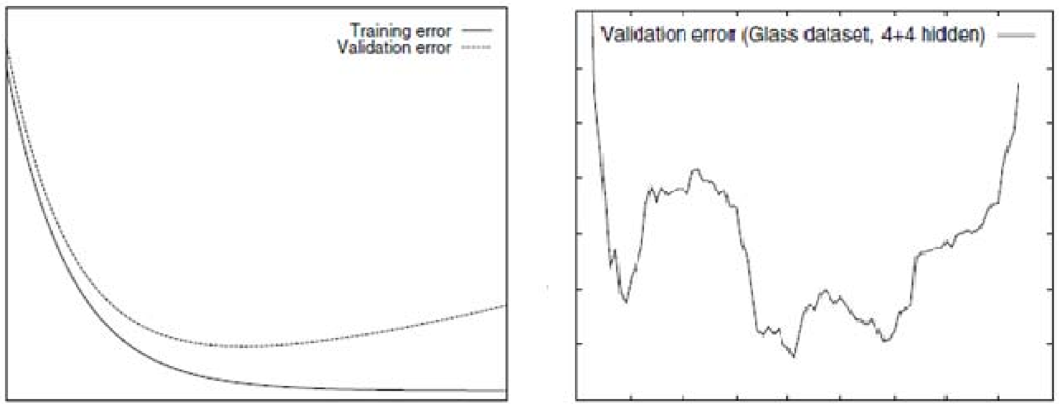


Figure 2-2 Theoretical vs Real World Validation Error

(Prechelt, 1998)

Figure 2-2 displays two separate graphs. For each graph, the horizontal axis represents training epochs and the vertical axis represents error. The graph on the left displays what validation error looks like in a perfect scenario. The graph on the right displays what validation error can actually look like (Prechelt, 1998). The scenario displayed in the graph on the right is much more likely to occur. Since this is the case, other stopping criteria have been developed. These types of stopping criterion include but are not limited to generalization loss threshold criterion, quotient of generalization loss and progress, and generalization increase over a given number of successive epochs. Prechelt tested these three methods and determined that the generalization loss method has the highest probability of providing "good" results.

2.6.1. Generalization Loss Threshold

$E_{opt}(t)$ is defined as the lowest validation set error obtained in epochs up to *t*.

$$E_{opt}(t) := \min_{t' \le t} E_{va}(t')$$

The generalization at epoch *t* is the relative increase of the validation error over so far:

$$GL(t) = 100 * \left(\frac{E_{va}(t)}{E_{opt}(t)} - 1 \right)$$

With the GL(t) value, the following stopping criterion is used:

GL_α: stop after first epoch t with GL(t)>α (Prechelt, 1998)

α is determined by the user. The reasoning behind this method is over fitting does not occur until error decreases slowly.

Instead of focusing on the validation error during training, this research will observe the activity of the upper layer weights to determine a termination point. The developed method will be compared to the generalization loss threshold stopping criterion.

2.7. Feature Selection

This section details different approaches used for feature selection, which can be thought of as a special case of architecture selection (Bacauskine & Verikas, 2004). Methods discussed include algorithms utilizing likelihood ratio test statistics and others using signal to noise ratios. The work done in feature selection, specifically the SNR saliency measure methods discussed in 2.6.3, is included because the application of these methodologies to hidden layer neurons is utilized in this research.

2.7.1. Likelihood ratio test statistics

Steppe, Bauer, and Rogers use likelihood ratio tests as part of a pruning algorithm and also use the same test statistic for feature selection. The likelihood test statistic is:

$$L = \frac{\frac{SSE_R - SSE_F}{df_R - df_F}}{\frac{SSE_F}{df_F}}$$ (Steppe, Bauer, & Rogers, 1996)

where F indicates the full model and R indicates the reduced model. The reduced model(s) possesses some combination of *k*-1 features where *k* features exist in the full model. The computational complexity of using this test statistic to determine the correct number of hidden layer nodes is much easier than feature selection. This holds true, because all combinations of *k*-1 features must be tested when one feature is being removed whereas the number of hidden nodes is simply decreased by one to test a smaller hidden layer structure.

2.7.2. Weight Based Saliency Measures

This section contains information regarding weight based saliency measures. They are of particular interest, because they are a building block for SNR saliency measures (discussed in 2.7.2). Previously developed feature screening methods based on saliency measures, such as the Belue-Bauer screening method and the Steppe-Bauer screening method, utilize a partial derivative based saliency measure and a weight based saliency measure (Belue & Bauer, 1995) (Steppe & Bauer, 1996). A weight based saliency measure, introduced by Tarr, is:

$$\tau_i = \sum_{j=1}^{J} (w_{i,j}^1)^2$$ (Tarr, 1991)

This measure determines the saliency of a feature by summing the squared values of the weights connecting feature *i* to the hidden nodes *j*. The superscript 1 indicates the weight is the first layer weight from node *i* to node *j*. This statistic is used because a square of the first layer weights in a salient feature will be significantly larger than a non-salient feature in an ANN.

2.7.3. Signal to Noise Ratio (SNR) Saliency measure

SNR Saliency measures are used to determine the relative value of features and provide an ability to rank features in order of importance (Ubeyli, 2008). The SNR saliency measure and the weight based saliency measure are similar in that they sum the squares of the first layer weights. However, the SNR saliency measure introduces an injected noise feature for comparison:

$$SNR_i = 10\log_{10}\left(\frac{\sum_{j=1}^{J}(w_{i,j}^1)^2}{\sum_{j=1}^{J}(w_{N,j}^1)^2}\right)$$

SNR$_i$ is the SNR saliency measure for feature *i* and $w_{N,j}^1$ is the first layer weight from noise node N to node *j*. J is the number of hidden nodes. The weights involving noise features are initialized and updated in the same way that weights from other features in the first layer are. Noise is injected with values that follow a Uniform(0,1) distribution. Further, the scaled logarithmic transformation of the ratio converts the saliency measure to a decibel scale (Ubeyli, 2008). Again, similar to the weight based saliency measure, the SNR saliency measure can be used to rank order the saliency of features where higher SNR saliency measure values correspond to higher feature saliency. The reasoning behind this SNR saliency measure is a relevant feature should have first layer weights emanating from it moving in a constant direction, as the ANN retrains, until the error is minimized. On the contrary, if a feature is not significant, the updates to the weights emanating from a feature's input node should be random and tend towards zero. With this idea, salient features should have a SNR saliency measure larger than 0 whereas non-salient features' SNR saliency measures should be close to or less than 0.0 (Bauer, Alsing, & Greene, 2000).

Belue & Bauer use a modified version of Tarr's saliency measure for feature screening to construct a model that reduces/eliminates non-salient features while maintaining generalization. Bauer, Alsing, and Greene's updated SNR feature selection method accomplishes this in a single training run with the SNR saliency measure and is summarized below (Bauer, Alsing, & Greene, 2000):

1. Introduce a Uniform (0, 1) noise feature *xN* to the original set of features.
2. Standardize all features to zero mean and unit variance.
3. Randomly initialize the weights between −0.001 and 0.001.
4. Randomly select the training and test sets.
5. Begin to train the ANN.
6. After each epoch, compute the SNR saliency measure for each input feature.
7. Interrupt training when the SNR saliency measures for all input features have stabilized.
8. Compute the test set classification error.
9. Identify the feature with the lowest SNR saliency measure and remove it from further training.
10. Continue training the ANN.
11. Repeat steps 6–9 until all the features (except the noise feature) in the original set are removed from training.
12. Compute the reaction of the test set classification error due to the removal of the individual features.
13. Retain the first feature whose removal caused a significant increase in the test set classification error, as well as
all features which were removed after that first salient feature.
14. Retrain the ANN with only the parsimonious set of salient input features.

This method is powerful in feature selection, because it can potentially only require a single training run. This is opposed to other screening methods, such as the Steppe-Bauer and Belue-Bauer methods that typically require 10 to 30 training runs

(Bauer, Alsing, & Greene, 2000). Ubeyli demonstrated the robustness of the SNR saliency metric in a study where he used the SNR saliency method to determine salient input features for a probabilistic neural network (PNN) used to classify internal carotid arterial Doppler signals (ICADS) (Ubeyli, 2008). After running the SNR saliency measure method, the PNN trained with only salient features outperformed the PNN using all original features as inputs. Ubeyli has also applied this same method to different data sets (Guler & Ubeyli, 2005).

2.7.4. Other Methods

Principal components analysis (PCA) is a classic approach to determining important variables in multivariate data. For more information regarding PCA see Multivariate Analysis: Methods and Applications (Dillon & Goldstein, 1984).

Cancelliere builds on the technique of PCA for feature selection. The method he presents performs PCA on every pattern presented by the features. After determining which features are non-salient, each exemplar’s value for the non-salient features is set to the average value of the respective feature across all exemplars (Cancelliere, 2003). This allows for a more efficient use of computation, because patterns in the data that are not important are avoided. This process is done during the training process and continues until classification accuracy increases significantly.

Setiano and Liu introduce an algorithm that selects features based on an augmented error function, which is used to identify salient features (Setiono & Liu, 1997). Following training, the network will have connections with large magnitude for features needed to represent underlying data patterns for classification.

2.8. General Ensemble Method

2.8.1. Introduction

The General Ensemble Method fuses network results together. The method uses information from a population of networks and combines them to provide the best possible linear combination.

As Perrone & Cooper present the method, they label the error for a network *i* as:

$$m_i(x) = f(x) - f_i(x)$$

This is the error between the desired output *f(x)* for a given exemplar and actual output for a given network *i*, $f_i(x)$. The general ensemble network can then be defined as:

$$f_{GEM}(x) = \sum_{i=1}^{i=N} \alpha_i f_i(x) = f(x) + \sum_{i=1}^{i=N} \alpha_i m_i(x)$$

where N is the total number of networks in the population and:

$$\sum \alpha_i = 1$$

These α_i values are calculated using a correlation matrix:

$$C_{ij} \equiv E[m_i(x) m_j(x)]$$

The relative error between a given network *i* and the error of the entire population of networks is then used to determine α_i:

$$\alpha_i = \frac{\sum_j C_{ij}^{-1}}{\sum_k \sum_j C_{kj}^{-1}} \qquad \text{(Perrone \& Cooper, 1992)}$$

This method is used to fuse results of network populations in this research to avoid encountering a matrix that has encountered a global minimum during its gradient search.

3. Methodology

3.1. Introduction

This section contains methodology information about two problems addressed in this research:

- building an efficient neural network middle layer structure
- determining when to conclude the training of a neural network

During the training process of a neural network, the weights connecting neurons in different layers fluctuate. This process occurs as the network attempts to correctly classify data input to the network. The work in this section is based on the intuition that the collection of weights emanating from a given middle layer neuron to each output neuron can provide valuable insight regarding the efficiency of network structure. By evaluating the collection of weights emanating from each middle layer neuron individually, it is possible that the contribution of adding a new middle layer neuron can be evaluated. Also, the amount of training time required can be estimated using information from these weights.

After using information from the upper layer weights to determine middle layer size and terminate training, the methodologies are combined with SNR feature selection to introduce a structured process for constructing a neural network.

3.2. Efficient Structure Using Upper Layer Weights

This section contains two different measures used in determining an efficient lower bound for middle layer structure. These measures are used in an algorithm proposed in 3.2.3.

3.2.1. Sum of Squared Weights (SSW)

The initial thought regarding the upper layer weights is they could provide insight regarding the optimal number of middle layer neurons. The training of the weights is a random process, so the value of an individual weight (i.e. a weight from a middle neuron *i* to an output neuron *j*) will vary from one training of a network to the next. The input into a given middle neuron is a single weight emanating from each of the features. While the weights emanating from a specific node will vary due to the randomness of the algorithm, if trained sufficiently, the network should classify data similarly from one training to the next. Using this logic, the sum of squared weights emanating from each middle layer neuron is recorded. For a given middle neuron *i* connected to upper layer neuron(s) *j*, this can be calculated as:

$$\sum_{j=1}^{J} (w_{i,j})^2$$

The sum of squared weights metric is chosen, because it rids the weight values of a sign and it also magnifies the weights with values larger than 1. The larger weights tend to have the biggest impact on classification (Sietsma & Dow, 1991).

3.2.2. Mean Sum of Squared Weights (MSSW)

Observing trends in the sum of squared weights (SSW) across all middle neurons led to a value of interest: the average SSW (MSSW) for a selected network structure. For a hidden layer containing *I* middle layer nodes, the MSSW can be calculated as:

$$MSSW = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}(w_{i,j})^2}{I}$$

Comparing the MSSW across different network structures led to an important observation. With the inputs and output layers held constant, as the number of middle layer neurons increase, the average MSSW will reach a maximum for a given middle layer structure and then steadily decrease as additional middle layer neurons are added. This can happen in two different ways.

3.2.3. MSSW Method

Using the observations discussed in section 3.2 thus far, an algorithm was constructed to reveal an effective baseline number of middle layer neurons. For future reference, MSSW$_x$ is the MSSW value for a network with x middle layer neurons. For a network with a selected number of inputs and outputs, the algorithm is:

1. Train 2 network structures:

   Network 1: Structure containing 1 middle layer node

   Network 2: Structure containing 2 middle layer nodes

2. If MSSW$_2$ > MSSW$_1$ continue to step 4. Otherwise go to step 3.
3. Let x denote the largest middle layer structure tested thus far. Train a new network structure with $x+1$ middle layer nodes. If MSSW$_x$ >MSSW$_{x-1}$, continue to Step 4. Otherwise, repeat.

*It is important to note that it is possible to get "stuck" in step 3 where $MSSW_x > MSSW_{x-1}$ does not occur. If this appears to be the case, select the first structure tested that provided a "significant" increase in validation accuracy

4. Let *x* denote the largest middle layer structure tested thus far. Train a new network structure with *x+1* middle layer nodes. If $MSSW_x < MSSW_{x-1}$, continue to Step 5. Otherwise, repeat.
5. Stop. The network configuration with *x-1* middle layer nodes is the effective baseline structure.

This algorithm proves effective as long as the network is trained sufficiently. Each time a network is trained it is recommended to train multiple times. In this research each network structure is run at least 5 times. This is done to provide insurance against the possibility of a neural network getting stuck in a local minimum while navigating the error surface during training. The reality of getting stuck in a local minimum on the error surface exists, because the starting point of the backpropagation algorithm is random in nature. Decisions based on the results of network trainings are made using the average values.

Figure 3-1 displays two examples of where the presented algorithm would terminate. In both plots, the algorithm would suggest 3 middle layer neurons.

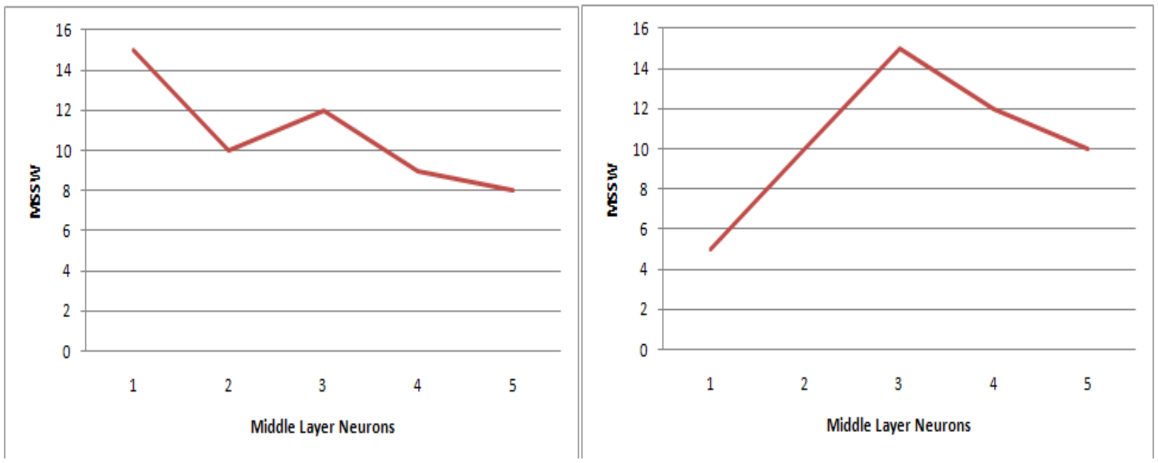


Figure 3-1 MSSW Example Graphs

3.3. Sufficiently Training an ANN

In section 3.2, the underlying assumption for the proposed algorithm is the network is sufficiently trained. If the network is undertrained, using an algorithm like the MSSW method can result in poor network performance. To analyze simulation output, it is important to remove transient data (Law, 2007). The goal in doing this is to remove bias created during a simulation model's warm-up period. Data remaining once the transient period is removed is known as steady-state data. Steady state can be analyzed to better understand the effects of the distribution(s) of the input data. There are multiple strategies attempting to accomplish this.

In an effort to integrate this practice to neural networks, the aforementioned truncation strategies were implemented on the training of neural networks. Throughout the training process, the upper layer weights of the network are altered. They initialize at some random point (in this research, between -.1 and .1) and take on different values based on the data input to the network. Initially, the intuition was the upper layer weights would stabilize at some point when the network reached a sufficient amount of training.

After reaching this point, the weight values would steady in the same way a mean value simulation does upon reaching the “steady-state” period. If this were the case, truncation methods used in the simulation realm could be used to determine a point where training is sufficient. Determining when the network reaches steady state would decrease time of training and ensure the network is not over trained.

3.3.1.   Sum of Weights

To start, the upper layer weights were observed individually to determine if they ever stabilized. This is not the case. Instead, they seem to tend in their respective directions. The next step in the process is to observe the sum of the upper layer weights.

$$\sum_i \sum_j (w_j^i)_k$$

Where $(w_j^i)_k$ is a weight from middle layer node *j* to upper layer node *i* during training epoch *k.*

3.3.2.   Incremental Change in Sum of Weights

The initial thought was the sum of weights for any network would tend toward zero during training. If this were the case, the intuition was once the sum of the weights approached zero, training could be terminated. Unfortunately this logic did not hold. The total sum of weights does not always approach zero. The next step was to observe the incremental change in the sum of the weights:

$$inc_k = \sum_i \sum_j (w_j^i)_k - \sum_i \sum_j (w_j^i)_{k-1} \forall k = 2, \ldots, Total\_Iterations$$

inc$_k$ is the incremental change from training epoch *k-1* to epoch *k.* The thought was *inc*$_k$ would decrease as a sufficient amount of training had occurred. After some initial testing, *inc*$_k$ does tend to get smaller as the training process unfolds. So, once the weight values start to "settle", training has concluded. However, the values of *inc*$_k$ where training should be terminated tend to vary based on both network structure and the magnitude of the input data.

3.3.3. Percentage Change in Sum of Weights

In attempt to make a general rule that could apply to multiple networks, the percentage change in sum of weights is calculated. For an epoch *k*, the percent change in the sum of weights is calculated as:

$$\%CHNG_k = 100 * \left| \frac{\sum_i \sum_j (w_j^i)_k - \sum_i \sum_j (w_j^i)_{k-1}}{\sum_i \sum_j (w_j^i)_{k-1}} \right|$$

The absolute value of the change is calculated because the direction of change was deemed unimportant.

3.3.4. Threshold stopping criterion

During the initial training of a neural network, the %CHNG value displays volatility. The first 500 to 1000 training epochs, depending on data being analyzed, tend to be unstable. This observation along with the %CHNG variable led to the proposed Threshold stopping criterion:

For any network structure, train for the minimum of:

1. 2500 Epochs
2. After 1000 epochs, the epoch where %CHNG < .025% for 50 consecutive training epochs

This method prevents the network from under training. The parameters used in this method were discovered by observation. For the data sets used in this research, 2500 training epochs appears to be the maximum amount required to train a successful network. Also, 1000 epochs is a good rule of thumb to prevent not training long enough. The %CHNG variable's threshold of .025% is another value determined through empirical observation. Paragraph 4.4 discusses an unsuccessful attempt to predict the %CHNG value necessary to terminate training for a given network structure/data set.

When using this method, a given network structure should be trained multiple times. The network structures being used in this research are relatively small. Due to the nature of the gradient search method used during back propagation, it is possible for the search to get halted in a local min when trying to minimize error (Bebis & Georgiopoulos, 1994). This can lead to poor performance. Any network structure should be trained multiple times to understand when this is occurring and avoid only having a network that is underperforming.

3.4. Combining Structure Selection, Feature Selection, and Training Termination Methods

Combining the MSSW method, Bauer, Alsing & Greene's SNR Feature Selection Algorithm (Belue & Bauer, 1995), and the proposed Threshold stopping criterion lead to a process for selecting a functional, efficient neural network structure. The process is summarized in Figure 3-2.

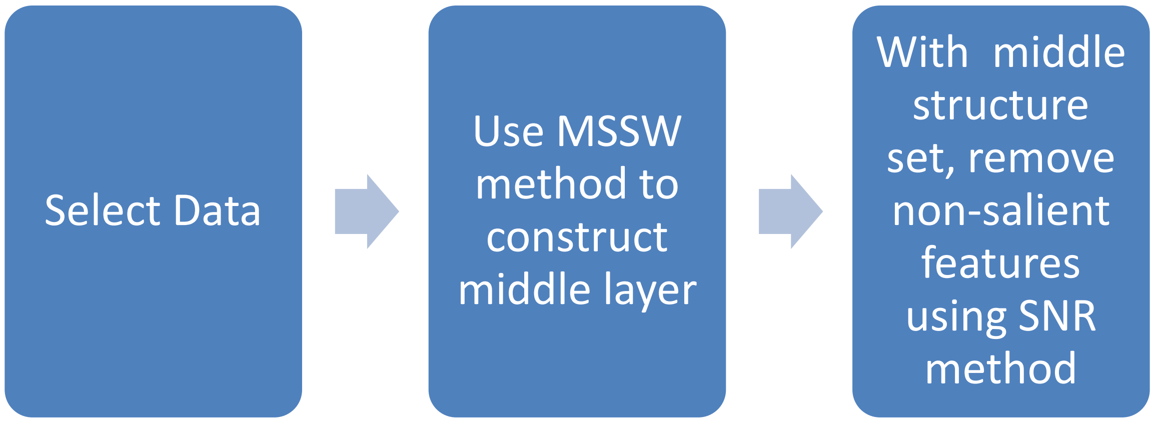

Select Data
Use MSSW method to construct middle layer
With middle structure set, remove non-salient features using SNR method

Figure 3-2 Combining Methods

When training a network use the Threshold stopping criterion. Also, when training a network, multiple replications are recommended due to stochastic nature of results. At least 5 replications are used in each case for this research, so average results can be observed.

A written summary of the Combined method:

Step 1:
Select Data
Step 2:
Implement middle layer selection algorithm (MSSW method) from section 3.2.
Step 3:
Implement SNR feature selection method from 2.7.3.
Step 4:
Stop.

During the training of any/all network configuration(s), use the proposed Threshold stopping criterion from section 3.3.4.

4. Results

4.1.Introduction

This chapter presents the results of applying the methods discussed in Chapter 3 to various data sets. After identifying the results of these methods, they are compared to results of the theories discussed in Chapter 2. Again these methods are building an efficient middle layer structure, terminating training, and combining the two with SNR feature selection.

4.2. Mean Sum of Squared Weights (MSSW)

The algorithm presented in 3.2.3 is applied to various data sets to show results for the building efficient network structure.

4.2.1. XOR Problem

The XOR problem in this research contains 150 data points, each with an *x* and *y* coordinate randomly generated by a Uniform(-1,1) distribution. The goal for the XOR problem is to classify the data points into two groups:

- Group 1- any point in quadrants 1 and 3 on the *x,y* coordinate plane
- Group 2- any point in quadrants 2 and 4 on the *x,y* coordinate plane

This means there are 2 input neurons and 2 output neurons. Other information about neural networks used:

- 5 replications of each network size are trained
- Each network is trained for 5000 training epochs
- 75% of the data is used as training data and 25% is withheld as validation data
- Step size (η), or learning rate, of .01 is used
- Lower layer weights are initialized randomly between [-.01.01]
- Upper layer weights are initialized randomly between [-.1, .1]

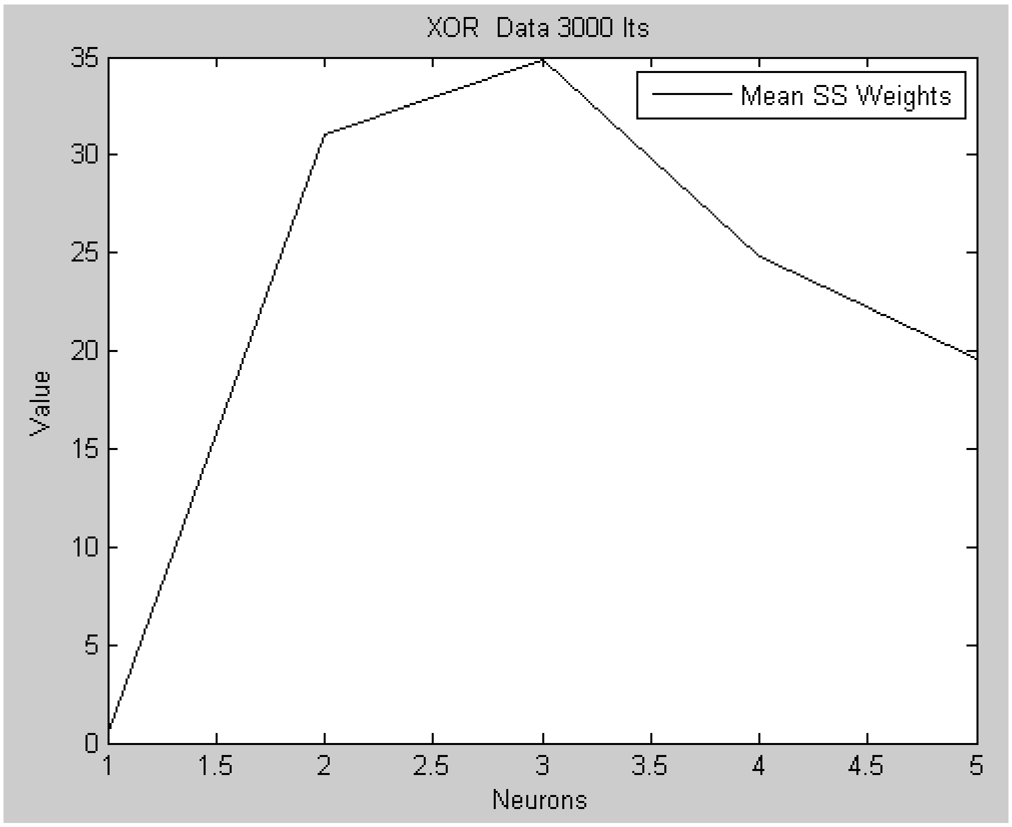


Figure 4-1 XOR Problem MSSW

Figure 4-1 displays that MSSW values for varying sizes of the network. The algorithm presented in 3.2.3 would terminate at 3 middle layer neurons. Figure 4-2 displays the accuracies corresponding to the different network structures tested in Figure 4-1.

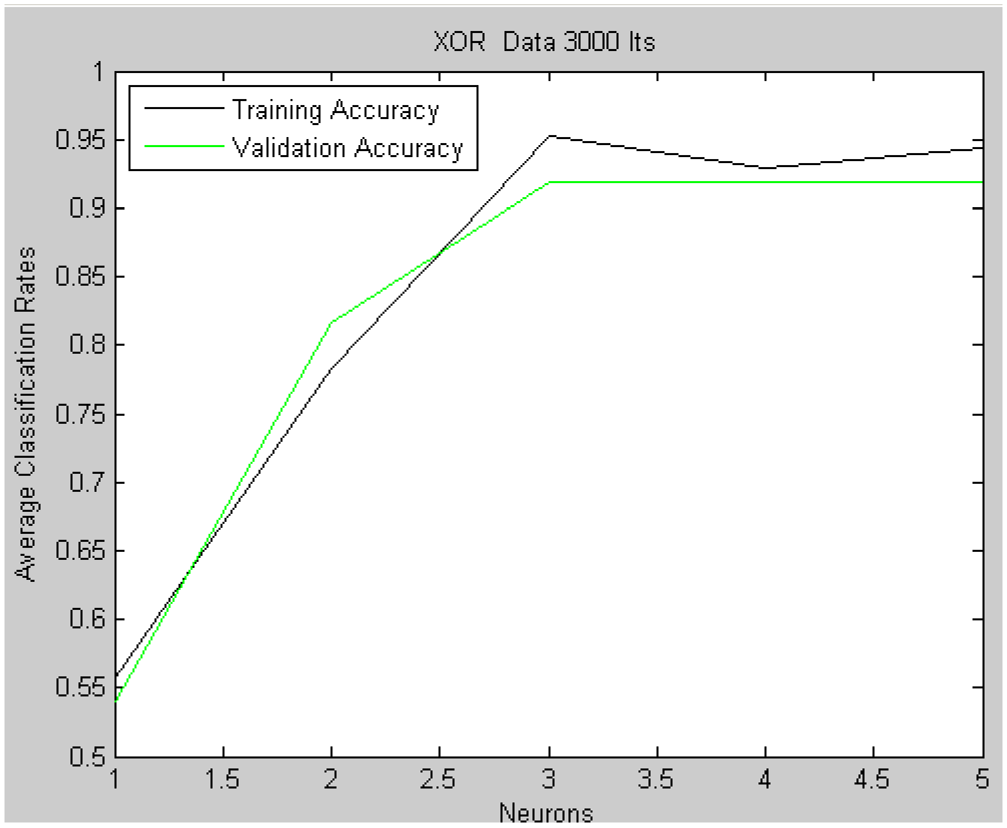


Figure 4-2 XOR Problem Accuracy

Figure 4-2 shows 3 middle layer neurons produce the highest validation accuracy. This implies that 3 middle layer neurons is not only a baseline, but could also be the optimal structure. At first this result seems odd. Simple logic implies that two lines could separate the data into their four separable regions, so only two hidden layer nodes are necessary. However, with only two middle nodes present, the network can get stuck in a local minimum while trying to navigate the error surface. When three nodes are present this is much less likely, which explains the increased performance from adding a third neuron to the middle layer (Sprinkhuizen-Kuyper & Boers, 1999).

Table 4-1 includes the results of the MSSW method and other empirical heuristic methods from section 2.5.

Table 4-1 XOR MSSW vs. Empirical Heuristics

Heurist Used	Middle Layer	Mean Training Accuracy	Stdev	Mean Validation Accuracy	Stdev
Kolmogorov	5	0.9434	0.0194	0.9189	0.0000
Lippmann	2	0.7823	0.0277	0.8162	0.1036
Zhang (LB)	3	0.9522	0.0048	0.9189	0.0000
Zhang (UB)	12	0.8779	0.0403	0.8595	0.0352
Daqi and Shouyi	4	0.9292	0.0000	0.9189	0.0000
Mirchandani&Cao	3	0.9522	0.0048	0.9189	0.0000
Gorman	2	0.7823	0.0277	0.8162	0.1036
MSSW	3	0.9522	0.0048	0.9189	0.0000

It appears that the MSSW performs as well as the other heuristics. Kolmogorov's heuristic results in higher validation accuracy, but the difference is minimal.

4.2.2. Body Fat Data Set

The data set contains 252 entries, each having 13 inputs:

- Age (years)
- Weight (lbs)
- Height (inches)
- Neck circumference (cm)
- Chest circumference (cm)
- Abdomen circumference (cm)
- Hip circumference (cm)
- Thigh circumference (cm)
- Knee circumference (cm)
- Ankle circumference (cm)
- Biceps (extended) circumference (cm)
- Forearm circumference (cm)
- Wrist circumference (cm)

These features are used to predict body fat percentage. In this problem, the neural network is used to determine if the entries are in one of two groups:

- Group 1: > 20% body fat
- Group 2: ≤ 20% body fat

The network has a structure with 13 input neurons and 2 output neurons with the following training parameters:

- 5 replications of each network size are trained
- Each network is trained for 5000 training epochs
- 75% of the data is used as training data and 25% is withheld as validation data
- Step size (η), or learning rate, of .01 is used
- Lower and Upper layer weights are initialized randomly between [-.1, .1]

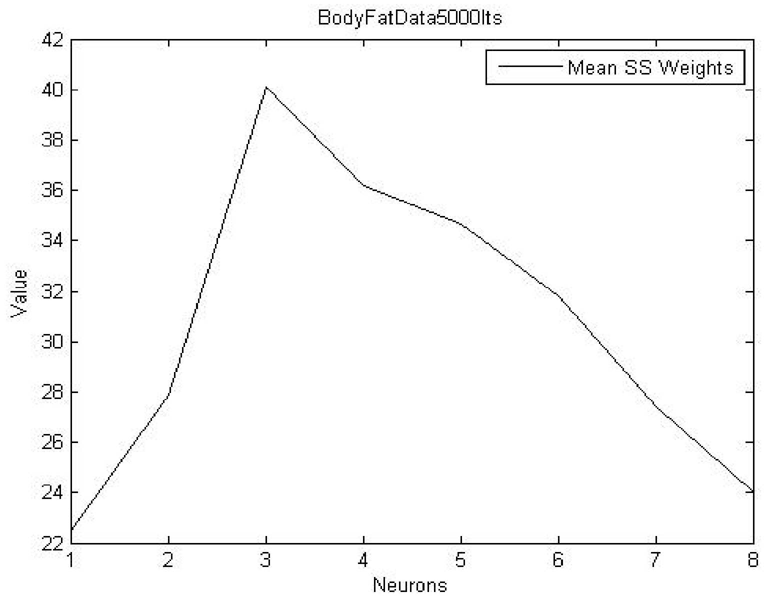


Figure 4-3 Body Fat Problem MSSW

Figure 4-3 displays the MSSW for the body fat data set while using network structures with an increasing number of middle layer neurons. The algorithm would terminate with 3 middle layer neurons. Figure 4-4 displays the accuracy of the network for the different network structures displayed in Figure 4-3.

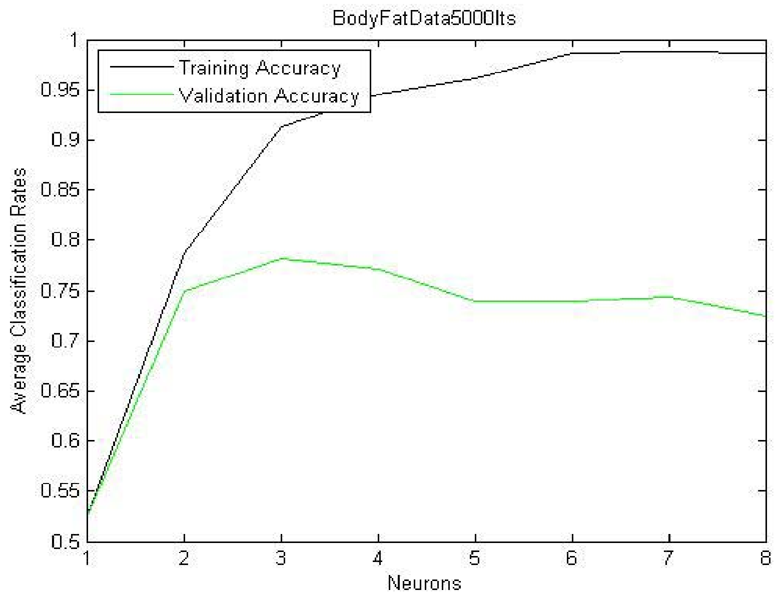


Figure 4-4 Body Fat Problem – Accuracy

Figure 4-4 shows that the highest validation accuracy occurs when 3 middle layer neurons are used. After this point, it appears that additional middle layer neurons result in overtraining (training accuracy increases while validation accuracy decreases).

In this instance, the MSSW method provides not only a good baseline network structure, but also the optimal structure. The results of the MSSW method are compared to other empirical heuristics discussed in chapter 2 in Table 4-2. The heuristics involving data patterns are exempt, because determining the patterns in a data set with a large number of features/outputs is beyond the scope of this research.

Table 4-2 Body Fat MSSW vs. Empirical Heuristics

Heuristic Used	Middle Layer	Mean Training Accuracy	Std. Dev.	Mean Validation Accuracy	Std. Dev.
Kolmogorov	27	0.9881	0.0090	0.7183	0.0456
Lippmann	6	0.9788	0.0106	0.7513	0.0330
Zhang (LB)	7	0.9868	0.0068	0.7183	0.0456
Zhang (UB)	16	0.9947	0.0075	0.7579	0.0152
Daqi and Shouyi	9	0.9881	0.0090	0.7381	0.0205
MSSW	3	0.9312	0.0106	0.7778	0.0449

It appears that the MSSW method performs better than the other structures tested. The recommended structure does not learn the training data as well as other structures, but the validation accuracy is the highest of any tested.

4.2.3. Finance Industry Data Set

The finance industry data set contains financial information about 25 companies across three industries (14 pharmacies, 5 textiles, and 6 super markets). There are seven features:

- Rate of Return (ROR)
- Debit/Equity Ratio (Deb/Eq)
- Sales
- Earning Per Share (EPS)
- Net Performance Margin (NPM)
- Price/Earnings Ration (P/E)
- Profitability

4 pharmacies, 1 textile, and 2 super markets are used for validation, while the rest are used for training. The structure of the network used is 7 inputs and 3 output neurons. Additional information about training:

- 5 replications of each network size are trained
- Each network is trained for 2500 training epochs
- 75% of the data is used as training data and 25% is withheld as validation data
- Step size (η), or learning rate, of .01 is used

- Lower layer and upper layer weights are initialized randomly between [-.1, .1]

The network is trained for 2500 training epochs , because observation of training runs revealed this to be a point where the network is not under-training.

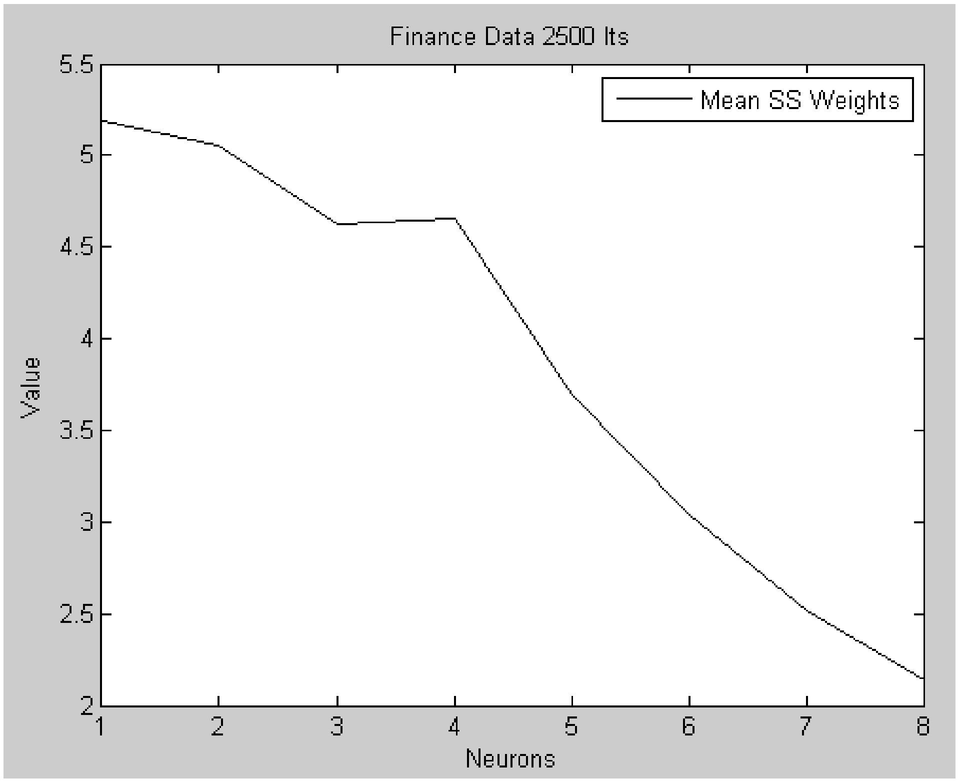


Figure 4-5 Finance Data MSSW

Figure 4-5, a plot of MSSW for different size networks, displays that the algorithm would terminate with 4 middle layer neurons.

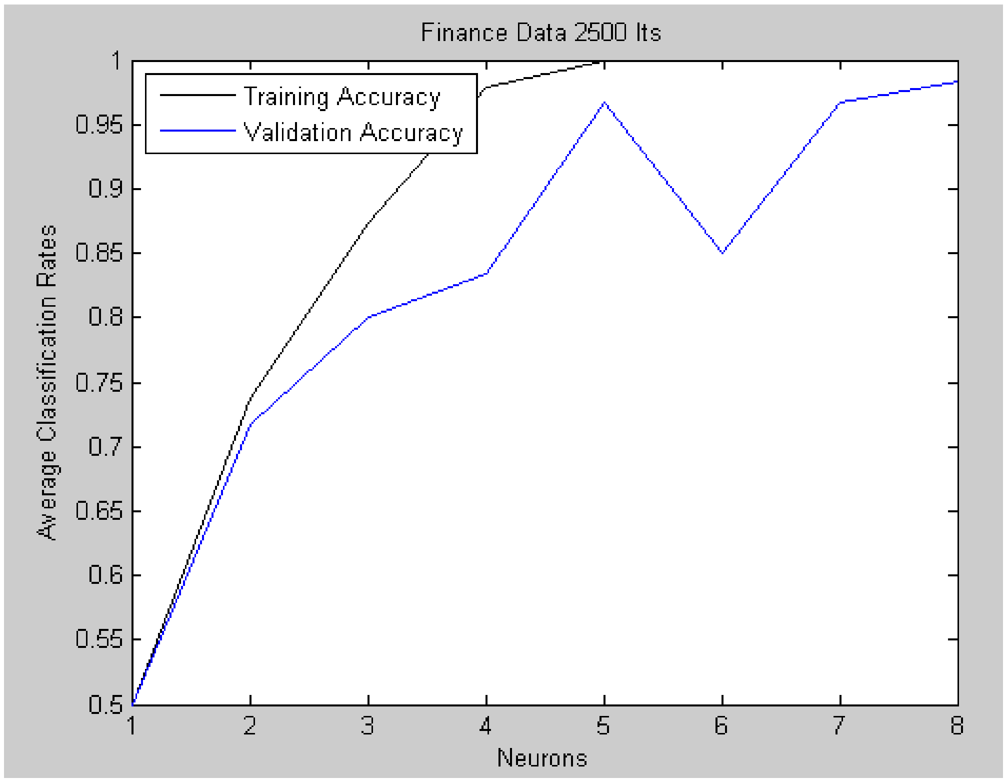


Figure 4-6 Finance Data Accuracy

Figure 4-6 shows that adding middle layer neurons, beyond the 4 recommended by the algorithm presented in 3.2.3, does increase the validation accuracy of the model. The hidden layer with 4 neurons provides an effective baseline structure, but not an optimal structure.

Table 4-3 contains the results of the MSSW method against some empirical heuristics discussed in section 2.5. The heuristics involving data patterns are exempt, because determining the patterns in a data set with a large number of features/outputs is beyond the scope of this research.

Table 4-3 Finance Data MSSW vs. Empirical Heuristics

Heuristic Used	Middle Layer	Mean Training Accuracy	Std. Dev.	Mean Validation Accuracy	Std Dev.	Average # Training Epochs
Kolmogorov	15	1	0	0.9667	0.0745	2500
Lippmann	5	1	0	0.9667	0.0703	2500
Zhang (LB)	6	1	0	0.8500	0.1657	2500
Zhang (UB)	15	1	0	0.9667	0.0745	2500
Daqi and Shouyi	7	1	0	0.9667	0.0703	2500
MSSW	4	0.97895	0.067	0.8333	0.1571	2500

The results in Table 4-3 are based on ten replications for each setting of the network. The results follow the conclusion after analyzing Figure 4-6. The MSSW method provides an effective structure, but not the optimal structure. The larger variance in validation accuracy between runs shows that there are times when the MSSW method works just as well as the other heuristics and there are times when it underperforms them.

4.2.4. Hot Dog Data Set

The next data set used is a hot dog data set. The data set contains 54 different hot dog brands which are made of three different meats (20 Beef, 17 “Meat”, & 17 Poultry). The data set containing the following features:

- $/oz
- $/lb protein
- Calories
- Sodium
- Protein/fat.

The goal of the neural network used is to predict the correct meat for each hot dog brand based on the above variables. The network used for the data has a structure of 5 input neurons and 3 output neurons. Additional information about training:

- 5 replications of each network size are trained
- Each network is trained for 2000 training epochs
- 75% of the data is used as training data and 25% is withheld as validation data

- Step size (η), or learning rate, of .01 is used
- Lower layer and upper layer weights are initialized randomly between [-.1, .1]

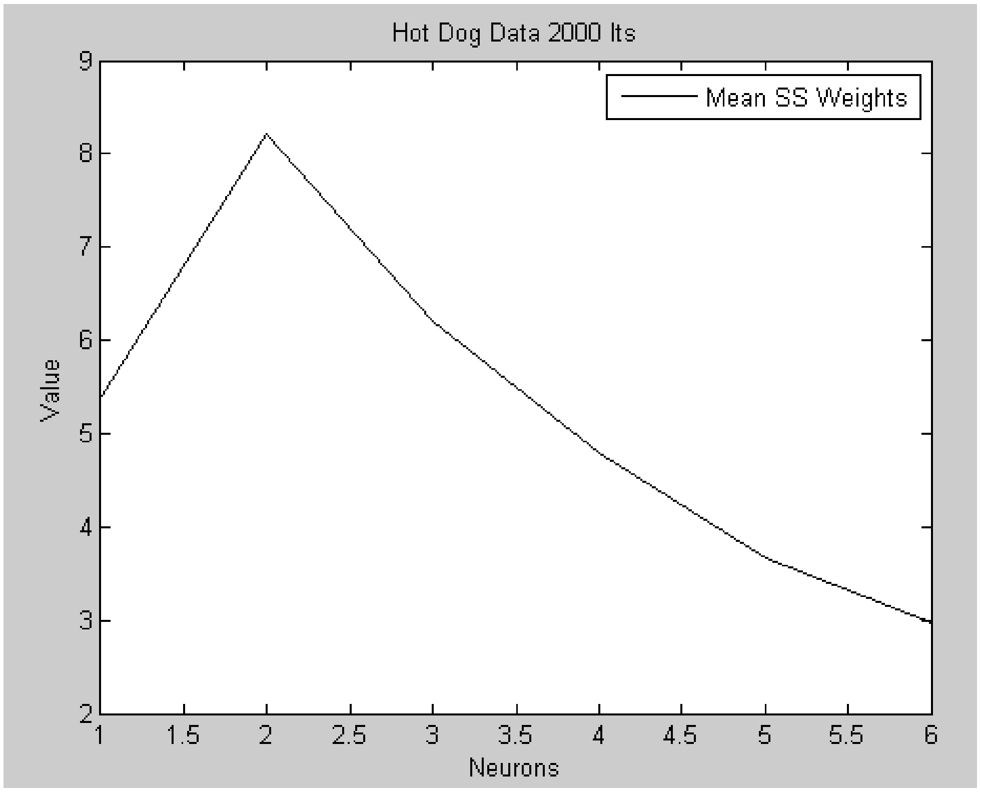


Figure 4-7 Hot Dog MSSW

The chart shown in Figure 4-7 shows that the maximum value of MSSW, which would terminate the proposed algorithm, occurs with 2 middle layer neurons.

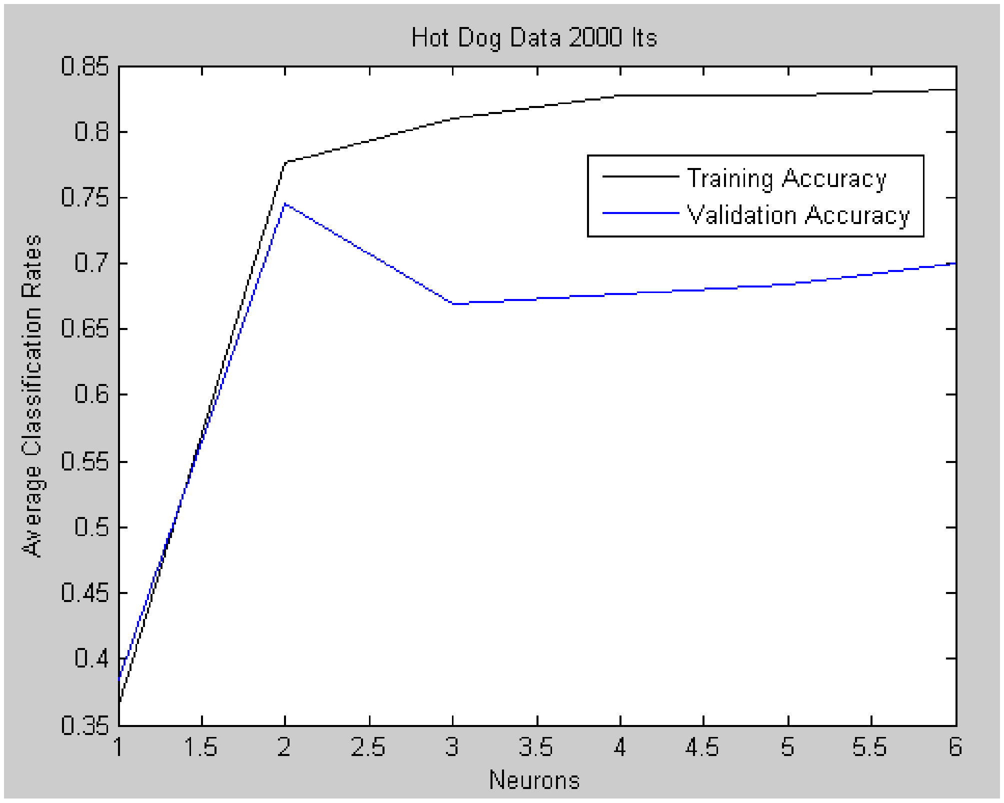


Figure 4-8 Hot Dog Accuracy

Figure 4-8 reveals that a network with 2 middle layer neurons works very well. The training accuracy is worse than any network structure with more than 2 middle layer neurons, while the average validation accuracy is the highest of any structure tested. The result achieved using the MSSW method is compared to other empirical heuristics in Table 4-4.

Table 4-4 Hot Dog MSSW vs. Empirical Heuristics

Heuristic Used	Middle Layer Neurons	Mean Train Accuracy	Std. Dev.	Mean Val Accuracy	Std. Dev.	Average # Training Epochs
Kolmogorov	11	0.9878	0.0172	0.6385	0.0519	2000
Lippmann	4	0.8268	0.0138	0.6769	0.0324	2000
Zhang (LB)	5	0.8268	0.0214	0.6846	0.0243	2000
Zhang (UB)	14	0.9780	0.0268	0.6538	0.1042	2000
Daqi and Shouyi	6	0.8317	0.0077	0.7000	0.0243	2000
MSSW	2	0.7756	0.0154	0.7462	0.0372	2000

The MSSW method performs better than any of the other heuristics tested. The method provides the highest validation accuracy and the lowest training accuracy.

4.2.5. Summary

This section applied the MSSW method to determine a successful lower bound for the middle layer of a neural network structure. The algorithm is used on four different data sets and proved successful in each case. Sometimes it appears that not only was the resulting middle layer structure an efficient baseline, but also optimal.

4.3. Sufficiently Training a Neural Network

4.3.1. Introduction

This section implements the proposed Threshold stopping criterion from section 3.3. The new method and other heuristics are applied to three different data sets. The structure used for each of the networks was recommended by the results from section 4.2. Using prior work to determine a successful structure is necessary so there is knowledge that training will eventually lead to an effective neural network. In section 4.5, the methods from 4.2 and this section are combined.

4.3.2. XOR Data Set/Discussion of Data Analysis

To first analyze the idea of terminating training, the XOR Data Set used in section 4.2.1 is evaluated. Again, the data set contains two inputs: an *x* and *y* coordinate. Any coordinate lying in the upper right or lower left quadrants of the *x*, *y* plane are considered to be in one class and the coordinates in the upper left and lower right quadrants are in another class. The following setup was used:

- 150 data points with x,y values generated from Unifrand(-1,1)
- 75% used for training (113)
- 25% used for validation (37)
- Network Structure: 2 Input Nodes, 3 Middle Layer Nodes, 2 Output Nodes
- 5500 Training Epochs
- 10 Runs (individual training of a network)
- Step size of .01
- Lower layer weights are initialized randomly between [-.01,.01]
- Upper layer weights are initialized randomly between [-.1, .1]

5500 training epochs are used, because after experimenting with the generalization loss heuristic, this seems like a good termination point. Section 4.2.1 displayed that successful classification can occur with less epochs. However, training past 5000 epochs could reveal there is room for improvement or maybe the network is already overtraining. The first step in analyzing the weights was to simply plot them for a single run and observe their activity. Figure 4-9 contains a plot of the weights over the course of training for a single run.

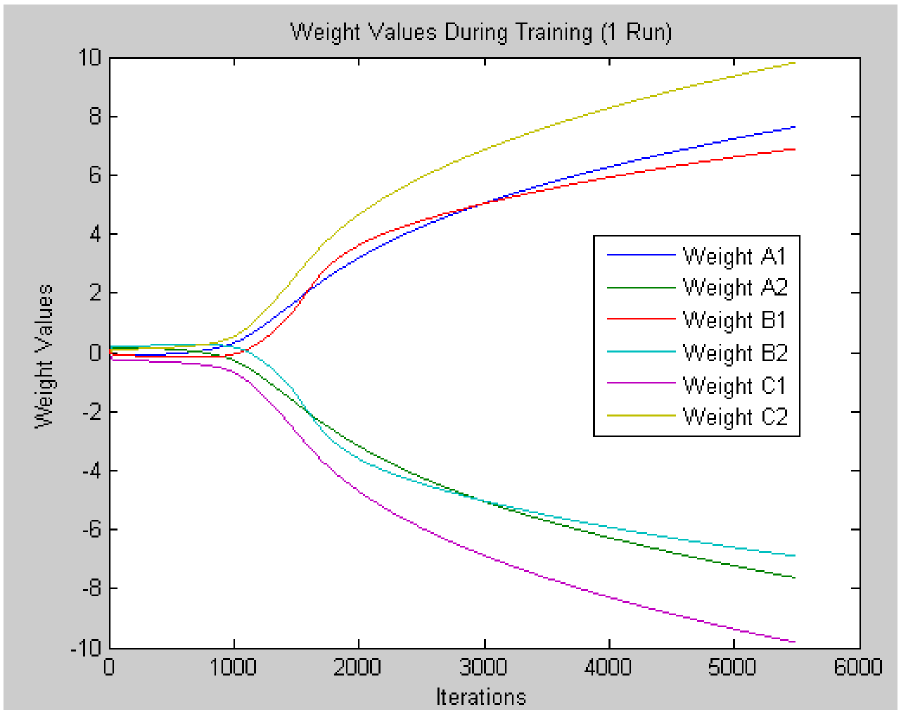


Figure 4-9 XOR Data Weight Values

The weights are labeled with their middle node of origin to the output node (i.e. middle nodes are A,B, &C and outputs are 1&2). Figure 4-9 reveals that the weights do not actually stabilize, but instead continually head in their respective directions. The activity of the weights brought forth the idea of looking at the sum of the weights over the course of training. For a given epoch, the sum of the weights would be calculated as follows:

$$\sum_i \sum_j (w_j^i)_k$$

Where $(w_j^i)_k$ is a weight from middle layer node *j* to upper layer node *i* during epoch *k.* This is plotted vs. the number of training epochs throughout the process of training.

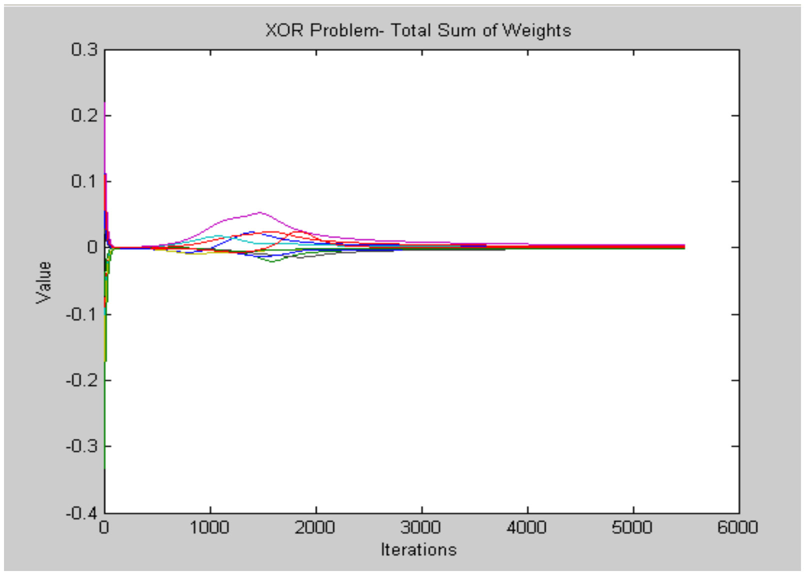


Figure 4-10 XOR Data Sum of Weights

Figure 4-10 shows the total sum of weights for all 10 runs of the network. There appears to be a lot of variation/volatility between 500 and 2500 training epochs, before the data all tends towards zero.

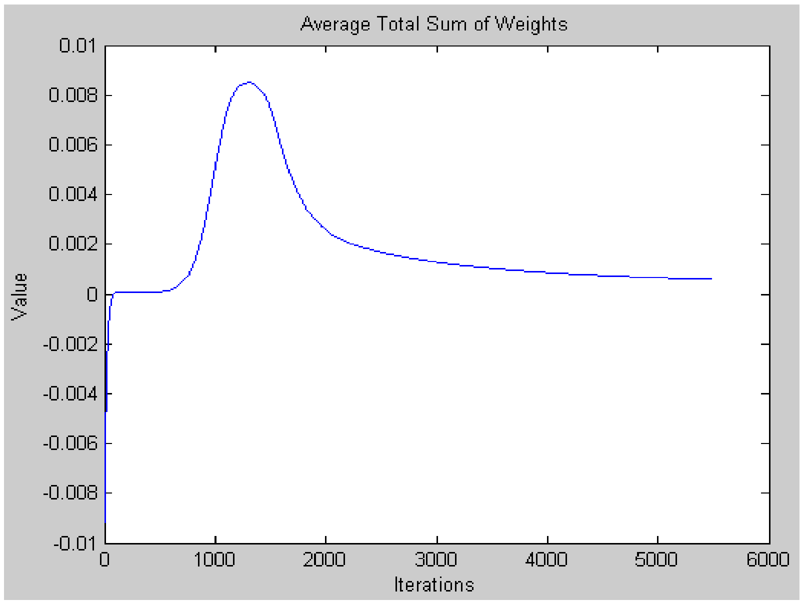


Figure 4-11 Average Sum of Weights

Figure 4-11 displays an average (across 10 replications) of the total sum of weights over the course of training. It should be noted that graph is more zoomed in on the y-axis than Figure 4-10 to better visualize the movement of the average throughout the training process. Unlike this problem, the sum of upper layer weights does not always tend towards zero, so the average will not always "converge" to the x-axis as it does in this problem. With this in mind, the change in the total sum of weights during the training process could provide insight regarding the termination of training. This incremental change can be calculated using the formula presented in 3.3.2:

$$inc_k = \sum_i \sum_j (w_j^i)_k - \sum_i \sum_j (w_j^i)_{k-1} \forall k = 2, \ldots, 5500$$

Again, $(w_j^i)_k$ is a weight from middle layer node *j* to upper layer node *i* during epoch *k.*

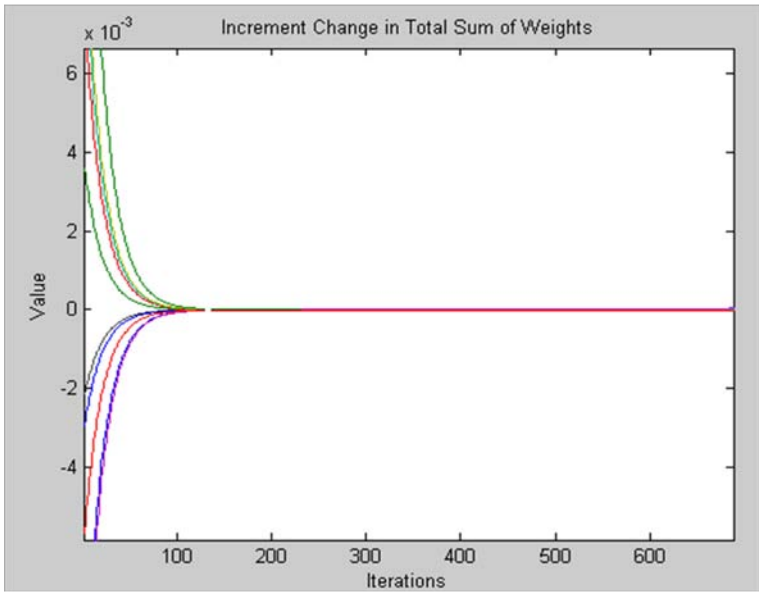


Figure 4-12 XOR Incremental Weight Change

Figure 4-12 is a plot of the incremental change in the total sum of weights across training epochs. Only the first 700 training epochs are shown, because the increments simply tend closer to zero for the remainder of training. It appears around training epoch 125 that the incremental change is reasonably small. The results of the network's ability to classify the data at this point are listed in Table 4-5:

Table 4-5 XOR Data Visual Incremental Truncation Accuracy

Results	
Train. Accuracy	**Val. Accuracy**
.5575	.5405

The results in Table 4-5 are the average across all 10 runs of the network. The network is slightly more accurate than a random guess, which is evidence that the network is undertrained at this point. Moving forward, the thought was to observe the percentage change in the sum of weights across training epochs. This is calculated as:

$$\%CHNG_k = 100 * \left| \frac{\sum_i \sum_j (w_j^i)_k - \sum_i \sum_j (w_j^i)_{k-1}}{\sum_i \sum_j (w_j^i)_{k-1}} \right| \forall k = 2, \ldots, 5500$$

The absolute value of the percentage change is used, because the direction of change is not of interest.

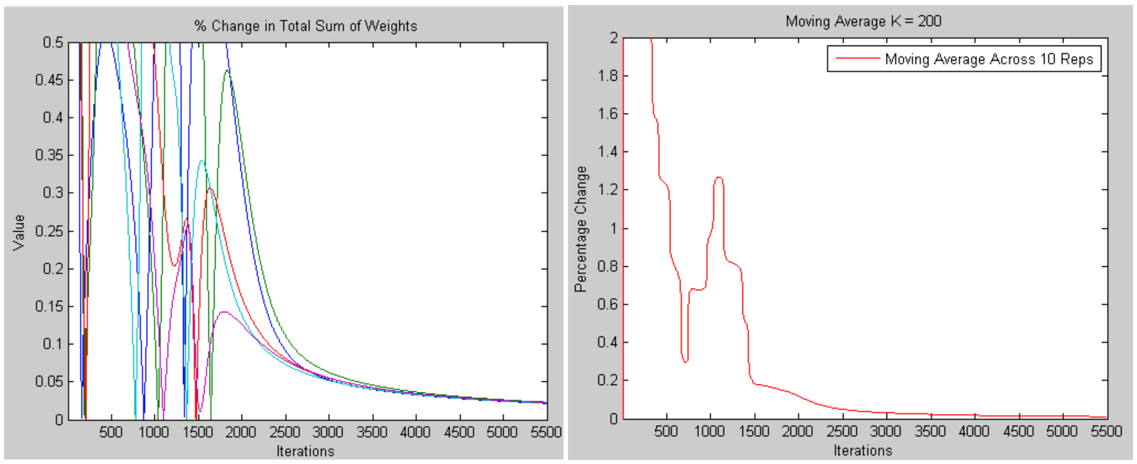


Figure 4-13 XOR Data %CHNG

Figure 4-13 contains a plot of the percentage change over the course of training for each rep individually (left) and a moving average (right). It appears that near training epoch 3000 all of the replications change close to the same amount for the remainder of training.

Welch's idea of visually choosing a point where data "converges" on a moving average graph can be applied to the upper layer weights of a neural network (Welch, 1981). Instead of choosing a point where the data becomes "steady-state", the point where the percentage change in the data substantially decreases in volatility is selected (3000 epochs). This strategy will be referred to as Welch's method for the remainder of this research.

Table 4-6 contains the results of stopping training after 3000 training epochs as opposed to the full 5500 training epochs that were run initially. After 3000 epochs, the network does a better job identifying validation data than at 5500 epochs, while not performing as well with the training data. This shows that stopping training at 3000 training epochs is beneficial, because between 3000 and 5500 training epochs the network is “over learning” the data.

Next, the Threshold stopping criterion from section 3.3.4 is used:

Table 4-6 XOR Data % CHNG Threshold Truncation Results

Method	Average # Training Epochs	Mean Train. Accuracy	Std. Dev.	Mean Valid. Accuracy	Std. Dev.
Threshold	2243.6	0.8938	0.0661	0.9027	0.0808
Welch	3000	0.9363	0.01	0.9189	0
Full	5500	0.9646	0	0.8919	0
Gen. Loss	5743.2	0.9646	0	0.8919	0

The Threshold stopping criterion appears effective for this data set. It requires the least training epochs and on average performs just as well as the other heuristics, but the results appear to be a bit less consistent. It is interesting to note that the generalization loss heuristic and terminating training after 5500 epochs provide the exact same performance every time.

4.3.3.   Discussion of Truncation Heuristics

Since all observed variations of the weights tend towards zero in the XOR problem, there appears to be no true “steady-state”. However, using Welch’s idea of visually choosing a point where data “converges” on a moving average graph displays the ability to provide successful results. Other heuristics for truncation which treat data as simulation output do not apply well, because they assume the data hovers around a true mean, which is not the case here.   Some of these heuristics include: SPC method, Randomization Test, Conway Rule, and the Crossing of the Means Rule. For information on these and other heuristics, see Mahajan and Ingalls’ survey paper (Mahajan & Ingalls, 2004).

4.3.4. Finance Data Set

After achieving some level of success with the XOR data set, the Threshold stopping criterion and Welch's method were applied to the Finance Industry data set used in 4.2.3. Again, this data set contains financial information about 25 companies across three industries (14 pharmacies, 5 textiles, and 6 super markets). There are seven input variables:

- Rate of Return (ROR)
- Debit/Equity Ratio (Deb/Eq)
- Sales
- Earning Per Share (EPS)
- Net Performance Margin (NPM)
- Price/Earnings Ratio (P/E)
- Profitability

The goal of this problem is to separate the data into the 3 separate industries. Other training information:

- 75% of data used for training
- 25% of data used for validation
- 10 Runs (individual training of a network)
- Step size of .01
- Initial weight values randomly assigned between ±.1

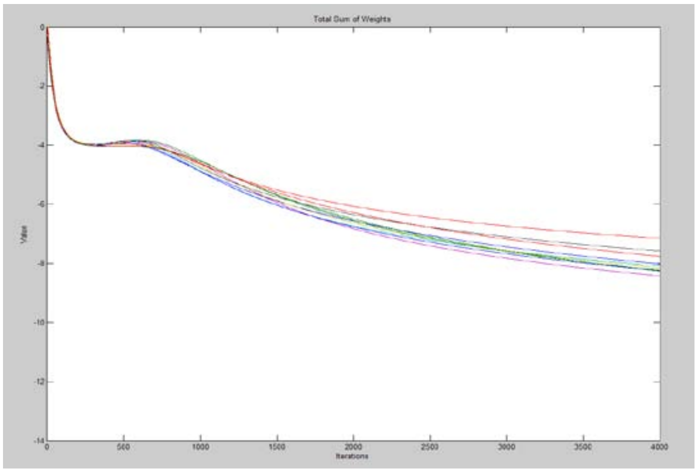


Figure 4-14 Finance Data - Total Sum of Weights During Training

Figure 4-14 displays the total sum of the weights during training for an individual replication of the network. As was mentioned previously, these weights do not center on the x-axis or zero, but instead become more negative as training continues.

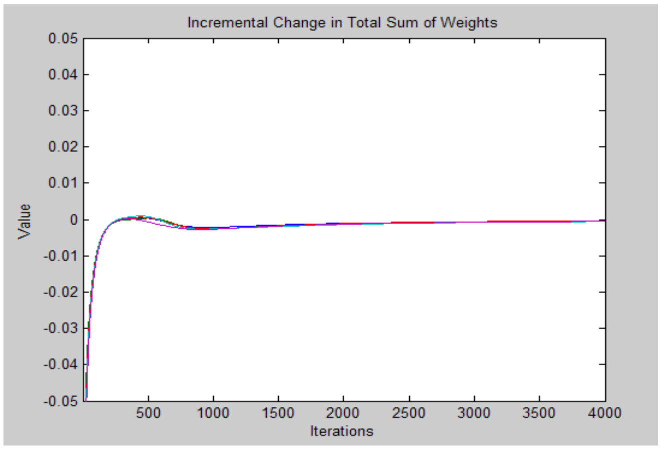


Figure 4-15 Finance Data - Incremental Change in Sum of Weights

Figure 4-15 is a plot of the incremental change of the sum of weights. This is similar to the XOR problem, because the data tend towards zero at the end.

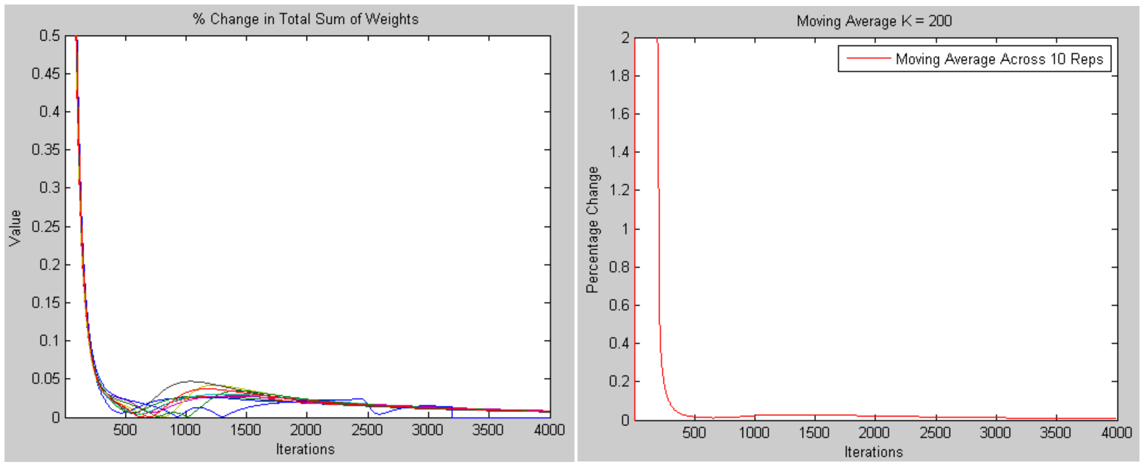


Figure 4-16 Finance Data - % Change in Sum of Weights

Figure 4-16 displays the percent change in the total sum of weights during training. The figure on the left has the values for all 10 runs plotted while the plot on the right has a moving average. Using the reasoning from the XOR problem that the transient period ends when the percent change settles close to zero, this data becomes stable around 1000 training epochs. Table 4-7 reveals the method of visual data truncation is not as successful for this problem. The proposed Threshold stopping criterion from 3.3.4 is also applied. For this data set, the criterion for stopping training is met on average at training epoch 1545.

Table 4-7 Finance Problem – Threshold Stopping Criterion Result

Method	Average # Training Epochs	Mean Train. Accuracy	Std. Dev.	Mean Valid. Accuracy	Std. Dev.
Welch	**1000**	0.8263	0.1165	0.6833	0.1459
Threshold	**1545.3**	0.8947	0.1265	0.8167	0.1657
Full	**4000**	1	0	0.9	0.1491
Gen. Loss	**6812.5**	0.9789	0.0666	0.8500	0.0946

Table 4-7 displays the network performance at this point. A generalization loss technique is also applied and proved to train much longer than expected. The generalization loss technique is run with α=1% and a max number of training epochs set at 7500. The average validation accuracy for the threshold stopping criterion appears successful, but displays a lot of volatility. The best truncation method appears to be stopping training at 4000 epochs.

4.3.5. Hot Dog Data Set

The Threshold stopping criterion is now applied to the hot dog data set presented in section 4.2.4. The network used for the data has a structure of 5 input neurons, 6 middle layer neurons, and 3 output neurons. Other training information:

- 75% of data used for training
- 25% of data used for validation
- 10 Runs (individual training of a network)
- Step size of .01
- Initial weight values randomly assigned between ±.1
- Trained for 5500 training epochs

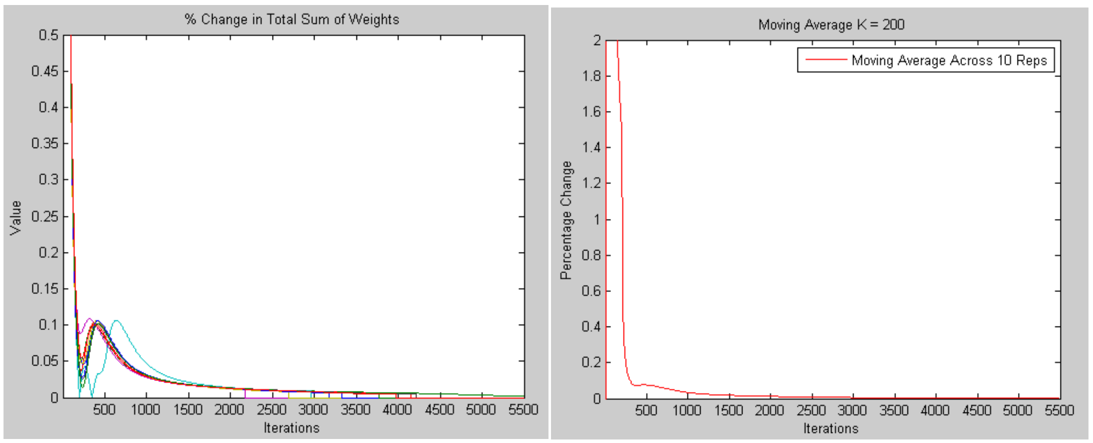


Figure 4-17 Hot Dog Sum of Weights

Figure 4-17 displays the percentage change in the total sum of the upper layer weights over the course of training. The plot on the right contains the moving average for this data. The plot of individual runs has a zoomed in y-axis to display the initial volatility. The truncation point using Welch's method is estimated at training epoch 800 and the Threshold stopping criterion terminates on average at training epoch 1490.

Table 4-8 Hot Dog Data Training

Method	**Average # Training Epochs**	**Mean Train. Accuracy**	**Std. Dev.**	**Mean Valid. Accuracy**	**Std. Dev.**
Welch	**800**	0.7537	0.0077	0.7846	0.0324
Threshold	**1210.7**	0.7659	0.0126	0.7615	0.0243
Gen. Loss	**3607.1**	0.8220	0.0231	0.6923	0.0000
Full	**5500**	0.9195	0.0258	0.7308	0.0831

Table 4-8 displays results of four different stopping criteria: Welch's truncation, Threshold, generalization loss, and 5500 training epochs. The generalization loss is run with an $\alpha=1\%$. As the network trains longer, it fits the training data better. It appears the validation accuracy decreases as training continues, which is expected. Each method tested appears to provide some tradeoff between average validation accuracy, training time, and volatility of results. Generalization loss provides the most consistent results, but the extension of Welch's method appears to provide the best overall results.

4.3.6. Body Fat Data Set

The truncation methods are applied to the body fat data set discussed in 4.2.2. Five separate trainings of the network determined by the MSSW method (3 middle nodes) are completed. Figure 4-18 plots each run's accuracy and %CHNG value against training epochs.

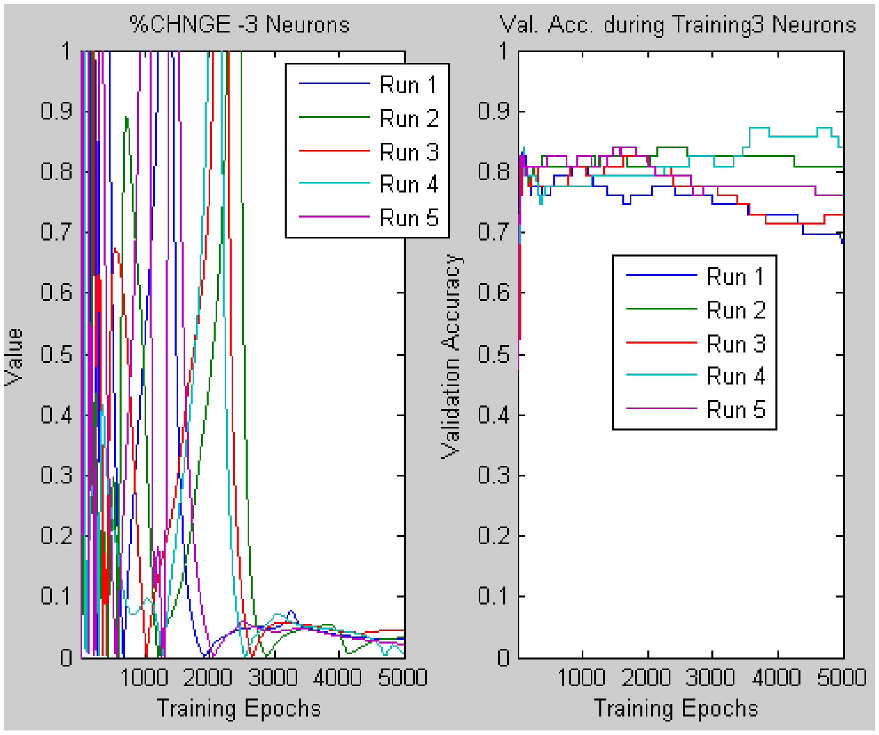


Figure 4-18 Body Fat Data %CHNG vs. Accuracy

In this example, the networks understand the data quickly. It appears, the sooner the network finishes training the better. The moving average of the %CHNG variable is visible in Figure 4-19.

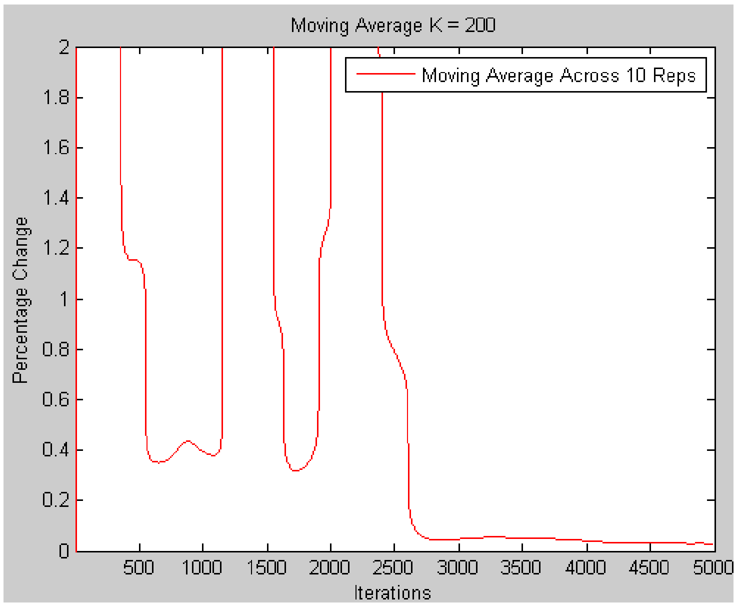


Figure 4-19 Body Fat Data Moving Average

The %CHNG value is quite volatile until about 2750 training epochs. Welch's method and other truncation method results are available in Table 4-9.

Table 4-9 Body Fat Data Terminating Criterion

	Average # Training Epochs	Mean Train. Accuracy	Std. Dev.	Mean Valid. Accuracy	Std. Dev.
Generalization Loss	1156.6	0.8720	0.0255	0.8032	0.0266
Threshold	1783.6	0.8444	0.0080	0.7873	0.0080
Welch	2500	0.9153	0.0179	0.7905	0.0325
Full	5000	0.9407	0.0177	0.7651	0.0629

It appears that the generalization loss method, using an α=1%, has the best performance. For the heuristics tested, it appears that shorter training periods provide better results.

4.3.7. Summary

In summary, there is information to be gleaned from observing the upper layer weights during the training process of a neural network. While training, the weights display a period of volatility before settling down. The longer the network trains, the less the sum of the weights changes. However, the weight values do not appear to settle at a mean value like output data in a mean value simulation. Instead, as the number of training epochs increases, the sum of the weights changes in decreasing increments. Percent change in the sum of the weights decreases during training as well.

The Threshold stopping criterion proposed in this research accounts for the initial volatility in the data and terminates training once the weights settle down. Using this method resulted in some under training of the network on the finance data set and was successful on the other ones tested. The generalization loss method was successful on all data sets, but on average required many more training epochs than the Threshold stopping criterion. Using Welch's method on the percent change in the sum of weights worked well on all but the finance data set. The network undertrained when analyzing the finance data set. It is probably a stretch to apply this heuristic to weight values, because they never truly settle down. Also, the results are very subjective since the method is dependent on a visual observation of when the data settles down. Due to the subjective nature of the Welch method results, the Threshold stopping criterion is the method of choice moving forward in the research.

4.4. Regression Equation for Stopping Training

4.4.1. Introduction

The goal in determining an efficient structure for a neural network is to decrease training time while still maintaining a network that generalizes well. Many factors affect the number of epochs required to sufficiently train. They include: number of exemplars, number of features, number of middle layer nodes, number of outputs, and the random nature of the back propagation algorithm. In an attempt to predict the number of epochs required in training, a regression on two different values was run. The two values of interest are:

- Value of %CHNG when max validation accuracy during training occurs
- Training epoch where max validation accuracy during training occurs

The regressions were not successful, but the results are available in Appendix A.

4.5. Combining Methods

4.5.1. Introduction

The goal in combining the middle layer selection algorithm, SNR feature selection, and upper layer weight training is to create an efficient, accurate neural network structure. Removing features and middle nodes decreases training time, while not necessarily decreasing the performance of the network. This section will apply the proposed Combined method to four data sets. The XOR data set is not included, because both features, the *x* and *y* coordinate, are essential to classifying data. Using the combined method on the XOR data set results in the same network/results as the Threshold stopping criterion did in section 4.3.2.

Again, the Combined method is summarized below:

Step 1:
    Select Data
Step 2:
    Implement middle layer selection algorithm (MSSW method) from section 3.2.
Step 3:
    Implement SNR feature selection method from 2.7.3.
Step 4:
    Stop.

During the training of any/all network configuration(s), use the proposed Threshold stopping criterion from section 3.4.

4.5.2. Metrics Used

Validation Classification Error – This is calculated as (1 - validation classification accuracy)

4.5.3. Fisher's Iris Data Set

Fisher's iris data set is a widely used multivariate data set (Bauer, Alsing, & Greene, 2000). This new data set is used to illustrate the combined method from start to finish without repeating information previously presented in this research. The data set has 3 classes of iris flowers: setosa, versicolor, and virginica. The data set contains four features: sepal length, sepal width, petal length, and petal width.

In order to execute the SNR method, a noise feature is also added to the data set. The noise feature is randomly generated from a uniform distribution between 0 and 1. The goal of this problem is to separate the data into the 3 separate classes. Other training information:

- 75% of data used for training
- 25% of data used for validation

- 10 Runs (individual training of a network)
- Step size of .01
- Initial weight values randomly assigned between ±.1
- Trained until Threshold stopping criterion is met

Step 2: Middle layer selection algorithm (MSSW method)

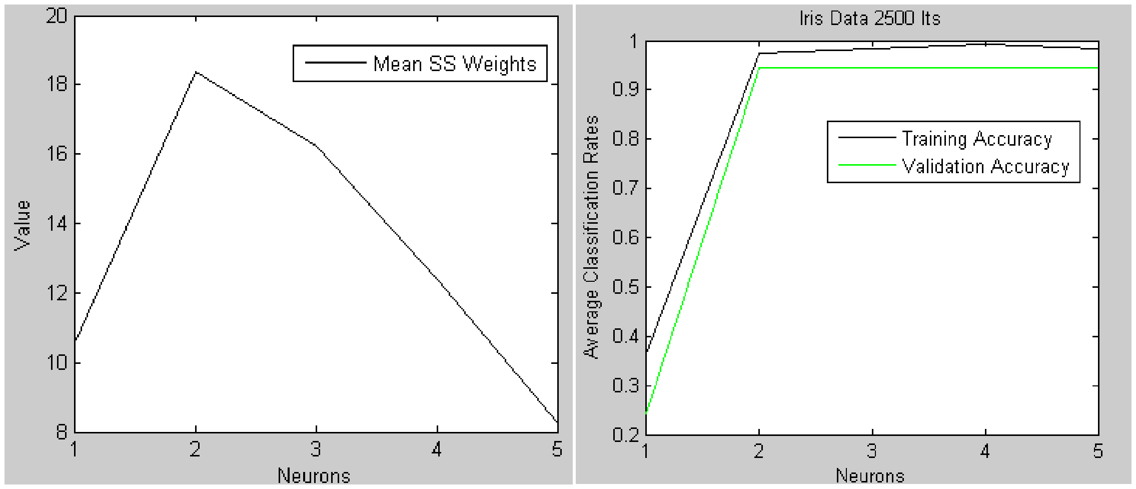


Figure 4-20 Iris Data all Features

For illustration's sake, this problem is run with 1,2,3,4, and 5 middle neurons. However, the MSSW middle layer algorithm would terminate at 2 middle neurons. It is visible in the accuracy chart that this point is ideal. However, 2 middle neurons provide an efficient baseline structure.

Step 3: SNR Feature Selection

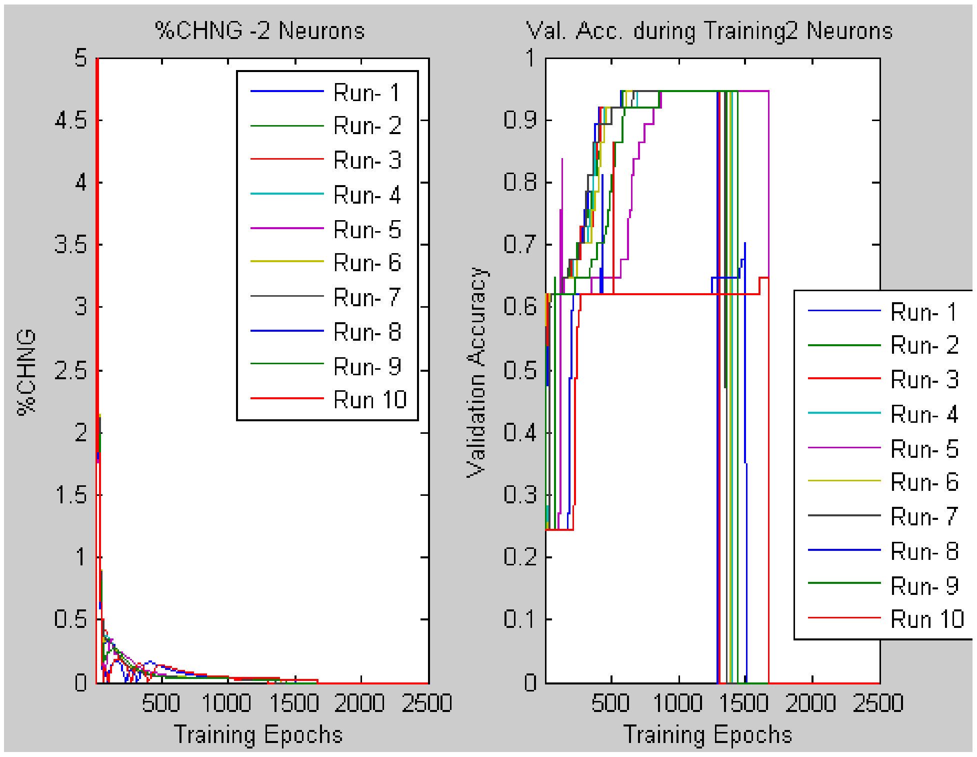


Figure 4-21 Iris Data All Features- 2 Middle Neurons

To start, the network is run 10 times with all features included. Figure 4-21 contains graphs of %CHNG and training accuracy through the course of training. The majority of the runs appear to classify at approximately 95% accuracy, but two seem stuck in a local minimum. Following the SNR feature selection algorithm, feature 1 is removed from the data set.

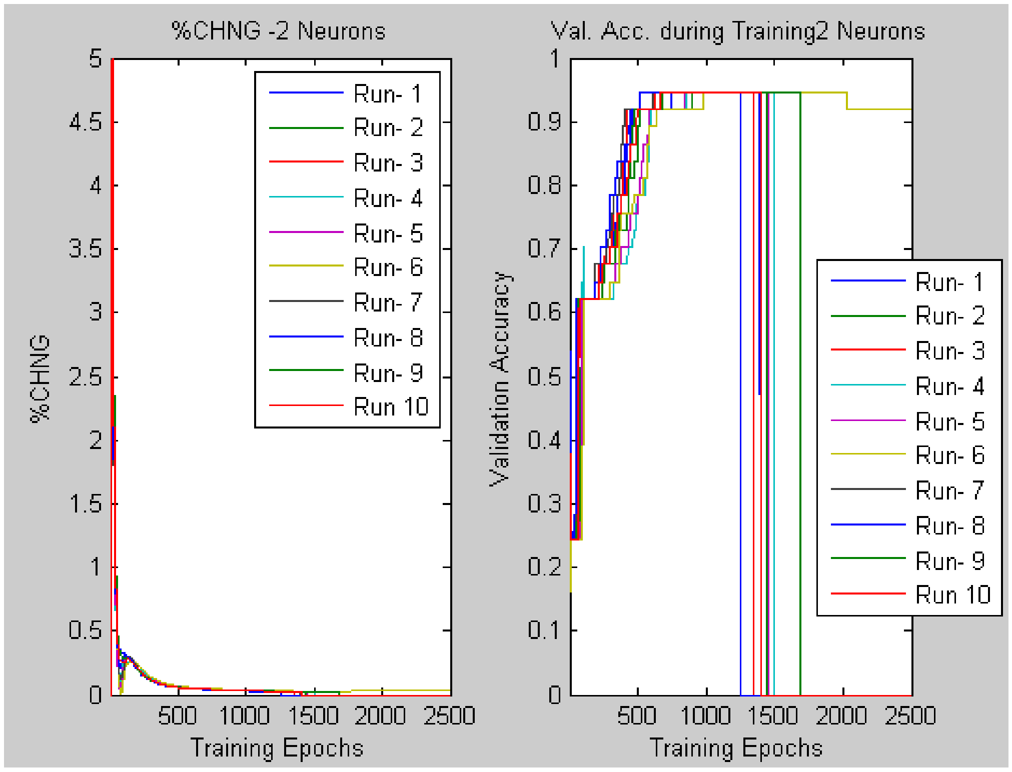


Figure 4-22 Iris Data- Feature 1 Removed

Removing feature 1 appears to rid the network of the problem of getting stuck in a local minimum. The SNR method recommends removing feature 2 for the next training epoch. Results of 5 runs at this setting are available in Figure 4-23.

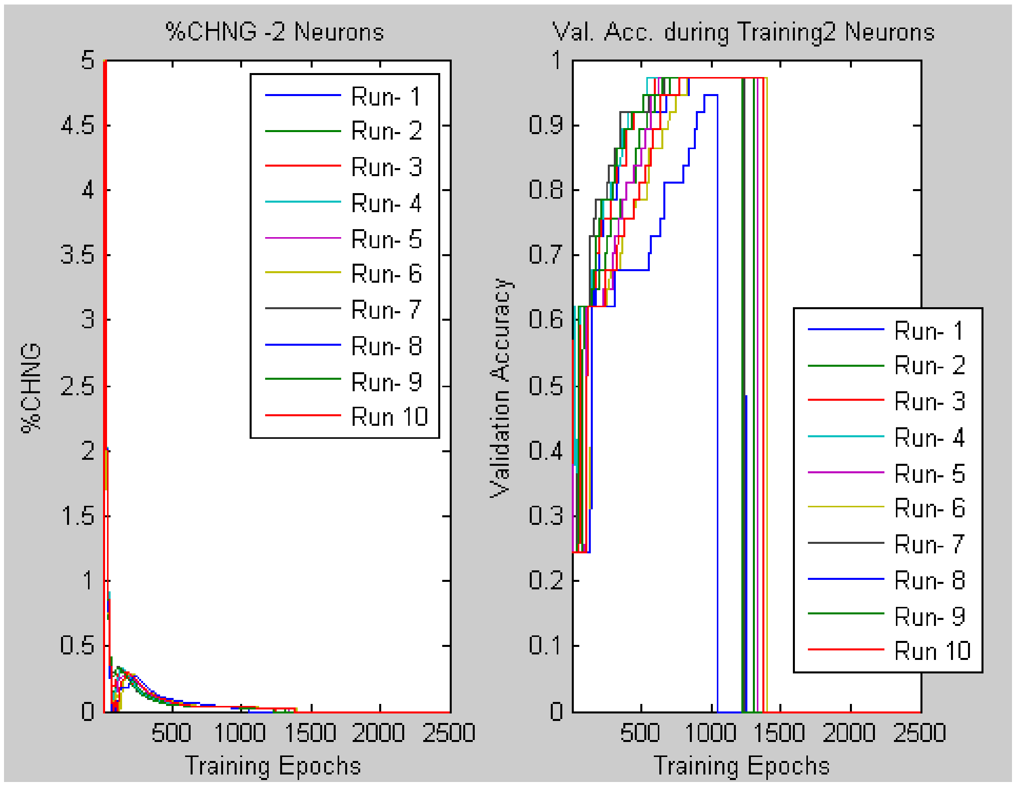


Figure 4-23 Iris Data- Feature 1&2 Removed

Removing feature 2 leads to the slightly higher accuracy while some runs train for fewer epochs. This improved the efficiency of the model. The SNR method now calls for the removal of feature 3.

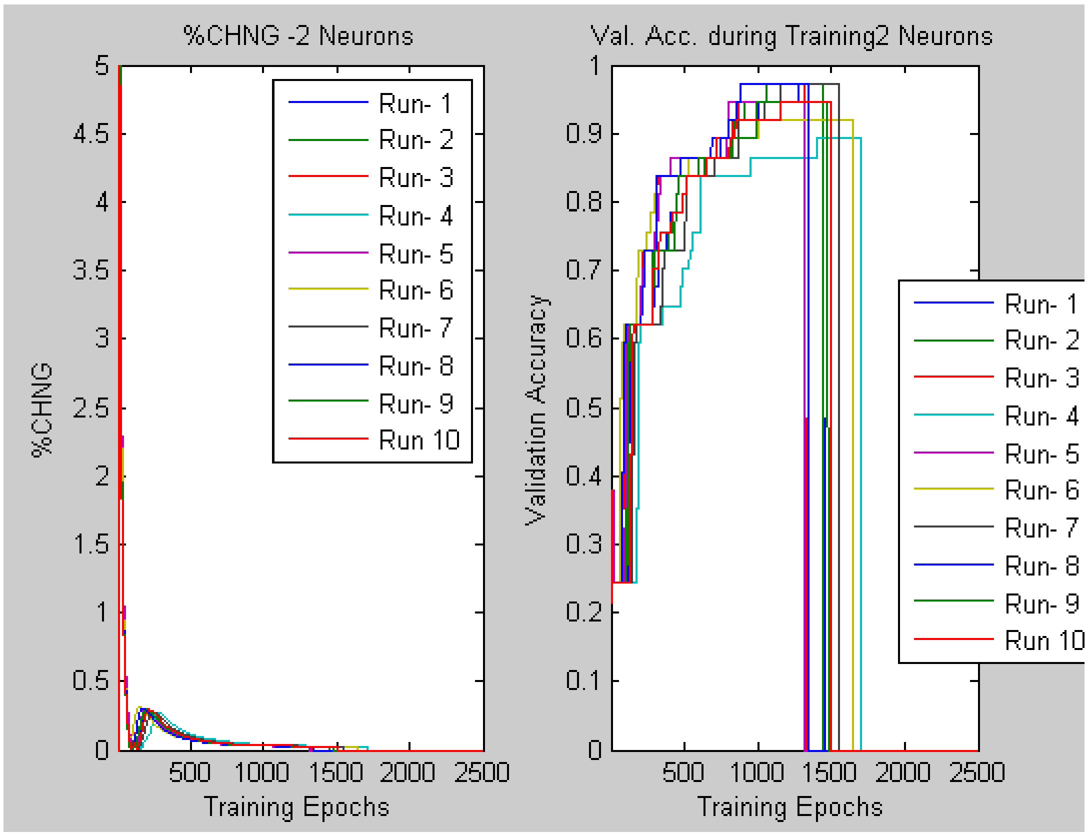


Figure 4-24 Iris Data- 1, 2& 3 Removed

The results appear very similar to the results of the previous network structure. This shows that the data is essentially a single variable problem. After removing the last feature, feature 4, there is a “significant” increase in the validation error. The validation error during the SNR feature selection process is plotted in Figure 4-25.

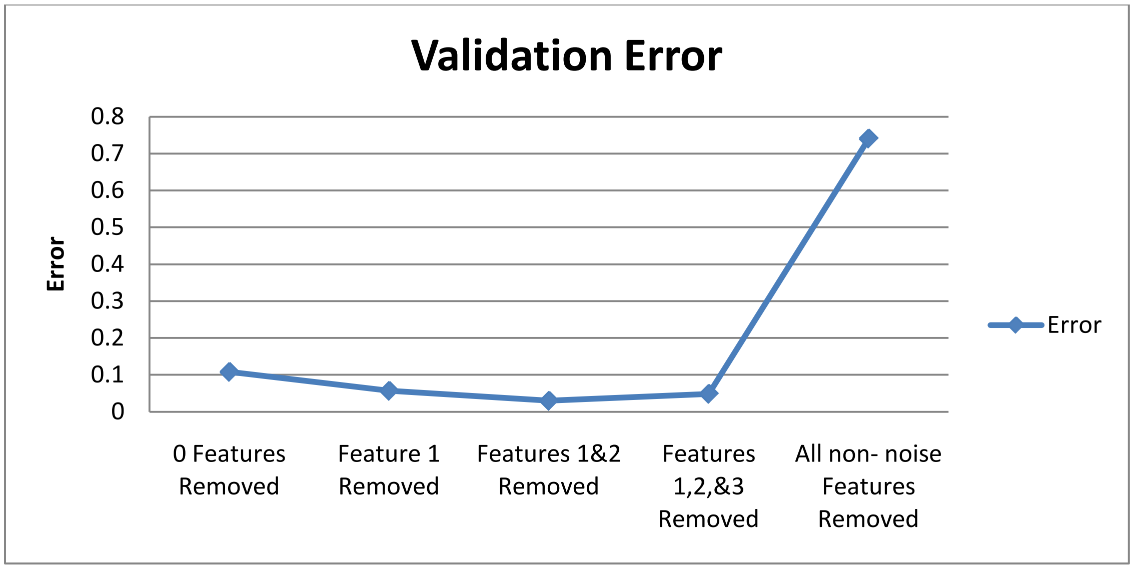


Figure 4-25 Iris SNR Validation Error

Feature 4, petal width, is the only salient feature, because there is a "significant" change in validation error when it is removed.

Using only feature 4 and the noise feature as inputs, the MSSW middle layer algorithm is run again. The results of the different feature sets used are summarized in Table 4-10. The selected feature set is shaded.

Table 4-10 Combined Methods Summary – Iris Data

	0 Features Removed	Feature 1 Removed	Features 1&2 Removed	Features 1,2,&3 Removed	All Features Removed
Mean Val Acc	0.8919	0.9432	0.9703	0.9514	0.2595
Mean Val Err	0.1081	0.0568	0.0297	0.0486	0.7405
Std Dev	0.1147	0.0085	0.0085	0.0279	0.0587

While not providing the best results, the recommended network has a smaller structure than the only network with better performance (network with features 1 & 2 removed), while requiring less training time. Overall, the combined method reduced the number of features from 4 to 1, while using only 2 middle layers neurons. This is a small, efficient network structure. Yang and Honavar provide results for the Cascade Correlation method for the iris data set based on ten runs.

Table 4-11 Cascade Correlation Method – Iris Data

Mean Val Accuracy	0.926
Std Dev Val Acc	0.014

The combined method outperforms the results displayed in Table 4-11. However, these results should be taken with a grain of salt, because Yang and Honavar do not specify how many data points were withheld for testing purposes (Yang & Honavar, 1991).

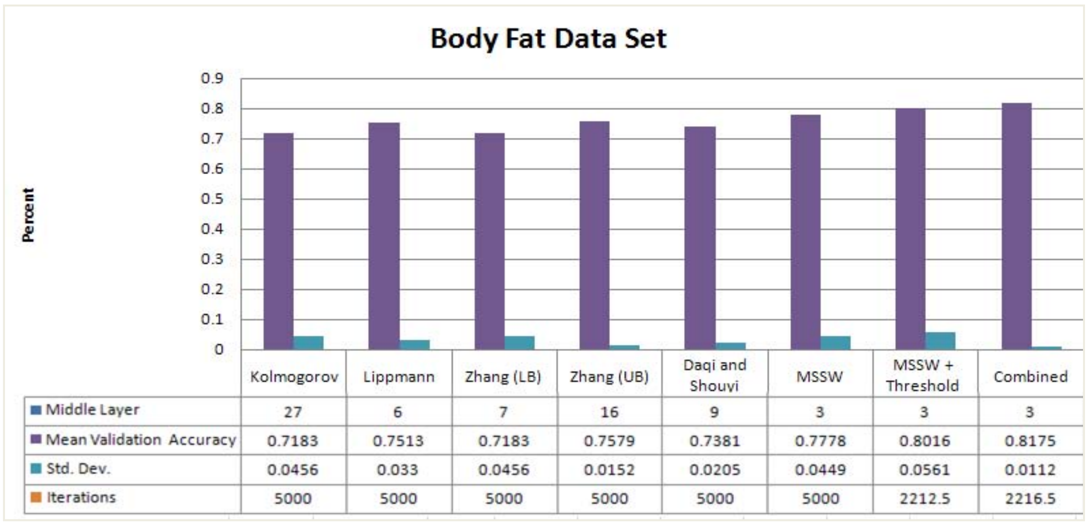
4.5.4. Body Fat Data Set

The combined method is now applied to the same body fat data set used in sections 4.2.2. and 4.3.6.. The following information is true for each network trained during the combined method:

- 75% of data used for training
- 25% of data used for validation
- 10 Runs (individual training of a network)
- Step size of .01
- Initial weight values randomly assigned between ±.1
- Trained until Threshold stopping criterion is met

Applying the combined method results in the following:

- Step 2 (MSSW method): 2 Middle Layer Neurons
- Step 3 (SNR feature selection): Remove all features but abdominal circumference



	Kolmogorov	Lippmann	Zhang (LB)	Zhang (UB)	Daqi and Shouyi	MSSW	MSSW + Threshold	Combined
■ Middle Layer	27	6	7	16	9	3	3	3
■ Mean Validation Accuracy	0.7183	0.7513	0.7183	0.7579	0.7381	0.7778	0.8016	0.8175
■ Std. Dev.	0.0456	0.033	0.0456	0.0152	0.0205	0.0449	0.0561	0.0112
■ Iterations	5000	5000	5000	5000	5000	5000	2212.5	2216.5

Figure 4-26 Body Fat Data – Combined Method

Figure 4-26 displays the results of the combined method against other heuristics previously discussed in the literature. In the figure, the validation accuracy is represented by the purple bars and the std. deviation by the teal bars. The combined method provides the highest validation accuracy with the lowest standard deviation across separate network trainings. The combined method clearly results in the model with the best generalizing ability. Table 4-12 summarizes the same data and includes information regarding accuracy of training data.

Table 4-12 Body Fat Data Combined Method vs. Other Heuristics

Heuristic Used	Middle Layer	Mean Training Accuracy	Std. Dev.	Mean Validation Accuracy	Std. Dev.	Average # Training Epochs
Kolmogorov	27	0.9881	0.009	0.7183	0.0456	5000
Lippmann	6	0.9788	0.0106	0.7513	0.033	5000
Zhang (LB)	7	0.9868	0.0068	0.7183	0.0456	5000
Zhang (UB)	16	0.9947	0.0075	0.7579	0.0152	5000
Daqi and Shouyi	9	0.9881	0.009	0.7381	0.0205	5000
MSSW	3	0.9312	0.0106	0.7778	0.0449	5000
MSSW + Threshold	3	0.8995	0.0150	0.8016	0.0561	2212.5
Combined	3	0.8095	0.0075	0.8175	0.0112	2216.5

4.5.5. Finance Data Set

The combined method is now applied to the same financial industry data set used in sections 4.2.3 and 4.3.4. The following information is true for each network trained during the combined method:

- 75% of data used for training
- 25% of data used for validation
- 10 Runs (individual training of a network)
- Step size of .01
- Initial weight values randomly assigned between ±.1
- Trained until Threshold stopping criterion is met

Applying the combined method results in the following:

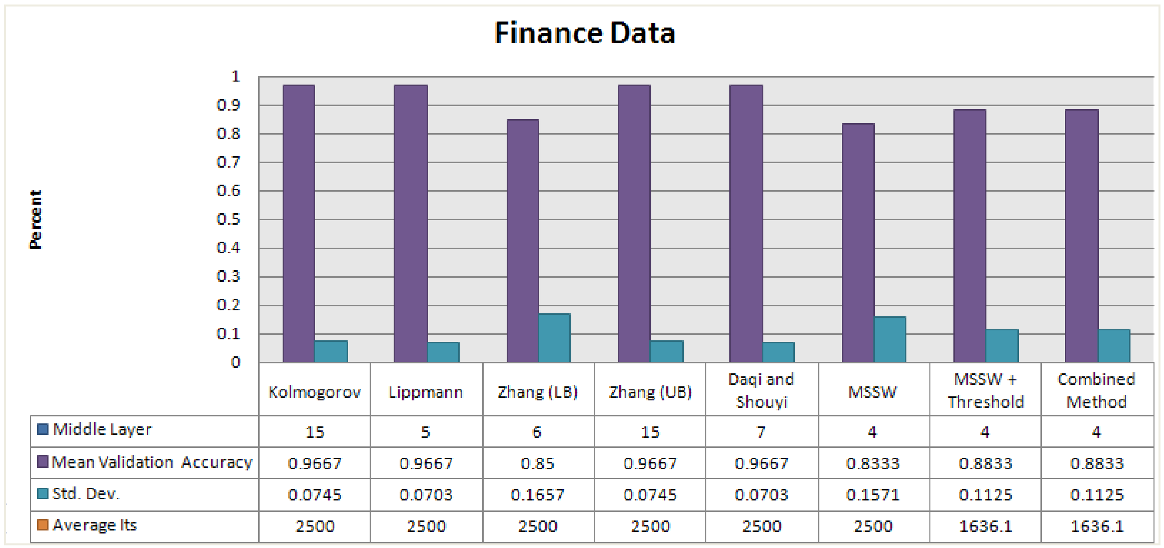
- Step 2 (MSSW method): 4 Middle Layer Neurons
- Step 3 (SNR feature selection): Remove no features

The resulting networks' performances from step 2 are documented in Table 4-13.

Table 4-13 Combined Methods Summary – Finance Data

	No Feature Removed	Feature 3 Removed	Features 3&7 Removed	Features 2,3 & 7 Removed	Features 1,2,3,&7 Removed	Features 1,2,3,6 & 7 Removed	Features 1,2,3,5,6 & 7	All Features Removed
Mean Valid. Acc	0.8833	0.75	0.85	0.7333	0.6833	0.5833	0.5333	0.5
Mean Valid. Error	0.1167	0.25	0.15	0.2667	0.3167	0.4167	0.4667	0.5
Std Dev.	0.1125	0.18	0.123	0.1405	0.123	0.1416	0.0703	0

The network resulting from the combined method includes all features along with 4 middle nodes. Figure 4-27 compares the results of this network structure to the structure containing all features while executing the MSSW method, the Threshold stopping criterion (MSSW+Threshold), and other heuristics analyzed earlier in this research.



	Kolmogorov	Lippmann	Zhang (LB)	Zhang (UB)	Daqi and Shouyi	MSSW	MSSW + Threshold	Combined Method
🟦 Middle Layer	15	5	6	15	7	4	4	4
🟪 Mean Validation Accuracy	0.9667	0.9667	0.85	0.9667	0.9667	0.8333	0.8833	0.8833
🟦 Std. Dev.	0.0745	0.0703	0.1657	0.0745	0.0703	0.1571	0.1125	0.1125
🟧 Average Its	2500	2500	2500	2500	2500	2500	1636.1	1636.1

Figure 4-27 Finance Data – Combined Method vs. Other Heuristics

Table 4-14 includes the same data as Figure 4-27 along with information about the training accuracy.

Table 4-14 Combined Methods Comparison – Finance Data

Heuristic Used	Middle Layer	Mean Training Accuracy	Std. Dev.	Mean Validation Accuracy	Std. Dev.	Average # Training Epochs
Kolmogorov	15	1	0	0.9667	0.0745	2500
Lippmann	5	1	0	0.9667	0.0703	2500
Zhang (LB)	6	1	0	0.85	0.1657	2500
Zhang (UB)	15	1	0	0.9667	0.0745	2500
Daqi and Shouyi	7	1	0	0.9667	0.0703	2500
MSSW	4	0.979	0.067	0.8333	0.1571	2500
MSSW + Threshold	4	0.9526	0.1006	0.8833	0.1125	1636.1
Combined Method	4	0.9526	0.1006	0.8833	0.1125	1636.1

The results of the combined method appear successful; however, additional structure in the hidden layer improves the performance of the network.

4.5.6. Hot Dog Data Set

The combined method is now applied to the same hot dog data set used in sections 4.2.4 and 4.3.5. Applying the combined method results in the following:

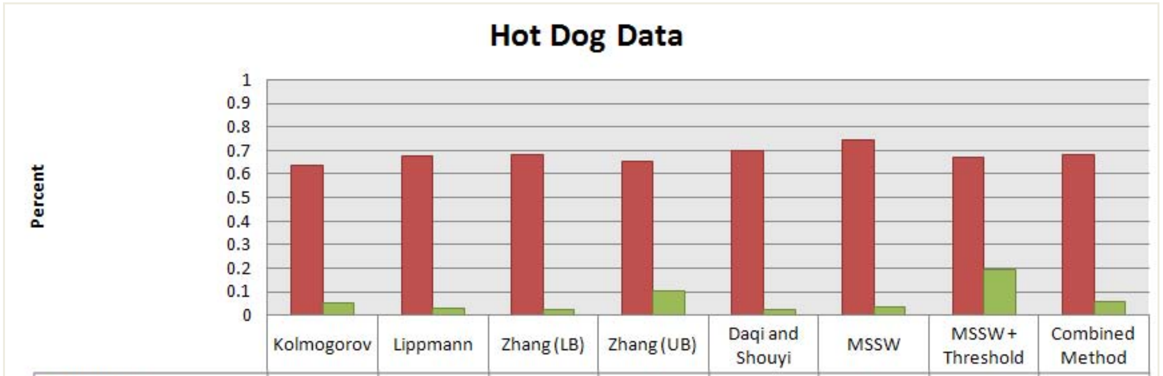
- Step 2 (MSSW method): 2 Middle Layer Neurons
- Step 3 (SNR feature selection): Remove features 5, 4, and 1

Each network structure is run five times. The resulting networks' performances in Step 3 are available in Table 4-15.

Table 4-15 Combined Methods Summary – Hot Dog Data

	No Features Removed	Feature 1 Removed	Features 5 & 1 Removed	Features 5, 4, & 1 Removed	Features 5, 4, 3, & 1 Removed	All Features Removed
Mean Valid. Accuracy	0.6692	0.7077	0.7154	0.6846	0.6	0.4
Mean Valid. Error	0.3308	0.2923	0.2846	0.3154	0.4	0.6
Std Dev.	0.1954	0.1486	0.1778	0.0568	0.1135	0.0873

The network structure with 2 middle layer neurons and only 2 features, $/lb protein and calories, is the structure chosen by the combined method.



	Kolmogorov	Lippmann	Zhang (LB)	Zhang (UB)	Daqi and Shouyi	MSSW	MSSW + Threshold	Combined Method
🟦 Middle Layer	11	4	5	14	6	2	2	2
🟥 Mean Validation Accuracy	0.6385	0.6769	0.6846	0.6538	0.7	0.7462	0.6692	0.6846
🟩 Std. Dev.	0.0519	0.0324	0.0243	0.1042	0.0243	0.0372	0.1954	0.0568
🟪 Average Its	2000	2000	2000	2000	2000	2000	1265.4	1362.4

Figure 4-28 Hot Dog Data – Combined Method vs. Other Heuristics

Figure 4-28 displays the results of the combined method against other heuristics tested thus far. The MSSW method with a full 2000 epochs appears to have the highest average validation accuracy and close to the smallest variance.

Table 4-16 Combined Method vs. Other Heuristics – Hot Dog Data

Heuristic Used	Middle Layer	Mean Training Accuracy	Std. Dev.	Mean Validation Accuracy	Std. Dev.	Average # Training Epochs
Kolmogorov	11	0.9878	0.0172	0.6385	0.0519	2000
Lippmann	4	0.8268	0.0138	0.6769	0.0324	2000
Zhang (LB)	5	0.8268	0.0214	0.6846	0.0243	2000
Zhang (UB)	14	0.978	0.0268	0.6538	0.1042	2000
Daqi and Shouyi	6	0.8317	0.0077	0.7	0.0243	2000
MSSW	2	0.7756	0.0154	0.7462	0.0372	2000
MSSW + Threshold	2	0.7147	0.1281	0.6692	0.1954	1265.4
Combined Method	2	0.6829	0.0736	0.6846	0.0568	1362.4

Table 4-16 includes the data from Figure 4-28 along with information regarding the training data. In the MSSW+Threshold stopping criterion, one of the ten replications yielded a validation accuracy of approximately .3 or 30%, which is the cause of the large validation accuracy variance. The advantage the combined method provides in decreased training epochs is countered by a decrease in validation accuracy. The MSSW method with a full 2000 training epochs provides the best network performance.

4.5.7. Summary

The combined method was tested on the iris, body fat, financial industry, and hot dog data sets. The networks resulting from the combined method have smaller structures than any other tested in this research. In the instance of the body fat and iris data sets the network structure appears successful. For the financial industry and hot dog data sets the performance results fall short of other methods tested. However, the drop off in performance could be worth the decrease in training time required by a smaller network. For the small data sets tested in this research this is likely not the case, but for a larger industrial data set, the tradeoff provided by the combined method could be more appealing.

4.6.Generalized Ensemble Method

4.6.1. Introduction

The generalized ensemble method (GEM) is introduced in this research to fuse multiple network results together to improve overall performance. In section 4.5 the combined method is applied to multiple data sets. Volatility is present in these results due to the stochastic nature of the backpropagation algorithm. Multiple networks are trained for each data set to display the variance of the results and provide insurance against the possibility of an individual run getting stuck in a local minimum during training. It would be possible to simply throw out a poor performing network. However, unsuccessful network trainings do not necessarily yield useless results. Using this train of thought, GEM is applied in this research to combine information from multiple networks. Further, GEM is proven to provide the best possible linear combination of a population of network's results.

4.6.2. GEM Application

To apply the GEM method, five separate runs of the networks recommended by the combined method are trained for each data set. The GEM method is then applied to combine the results of these five network trainings.

Table 4-17 GEM Application

	Combined Method Result		GEM
Data Set	**Avg. Validation Accuracy**	**Std Dev.**	**Validation Accuracy**
XOR	0.9027	0.0808	0.9189
Body Fat	0.8175	0.0112	0.8254
Finance	0.8833	0.1125	1
Hot Dog	0.6846	0.0568	.6923
Iris	0.9514	0.0279	0.973

Table 4-17 displays the GEM validation accuracy is higher than the average of the combined method runs for all five data sets. For each data set, the GEM results are close to if not the best results displayed thus far for any heuristic in this research.

5. Conclusion/Future Work

5.1. Conclusion

During this research, the analysis of upper layer weights in feed-forward neural networks led to a proposed method to choose a baseline structure (MSSW method), a proposed method to terminate training (Threshold stopping criterion), and an iterative process to arrive at an efficient neural network structure (Combined method).

The motivation in analyzing the upper layer weights was to discover an optimal middle layer structure. Unfortunately, the MSSW method is not always optimal. With that said, neither are any of the heuristics tested in this research. For the data sets tested, the MSSW method appears to provide a reliable baseline, or lower bound, for effective structure, and outperformed empirical heuristics more times than not.

Observing the activity of the upper layer weights also led to the Threshold stopping criterion for terminating training. This criterion does cut off the training prematurely at times, but this is partially due to the nature of feed forward neural networks. The backpropagation gradient search algorithm can get stuck in local minimums when using smaller middle layer structures.

Combining the MSSW method and the Threshold stopping criterion with SNR feature selection provides a decrease in training time. However, the decrease in training time does come at a cost. By removing "non-salient" features from the model, performance tends to decrease and become more volatile. Taking this method one step further and applying the general ensemble method removes this volatility from the results and can yield even higher validation accuracy.

5.2. Future Work

The current process of the MSSW method requires multiple trainings of each network size. Conducting this process in a single training run, by adding hidden layer neurons during training, would decrease time to perform the MSSW method.

Appendix A.

Regression for stopping criterion

The two values of interest are:

- Value of %CHNG when max validation accuracy during training occurs
- Training epoch where max validation accuracy during training occurs

These two values were recorded for multiple data sets and structures. The different networks structures tested are listed in the table below.

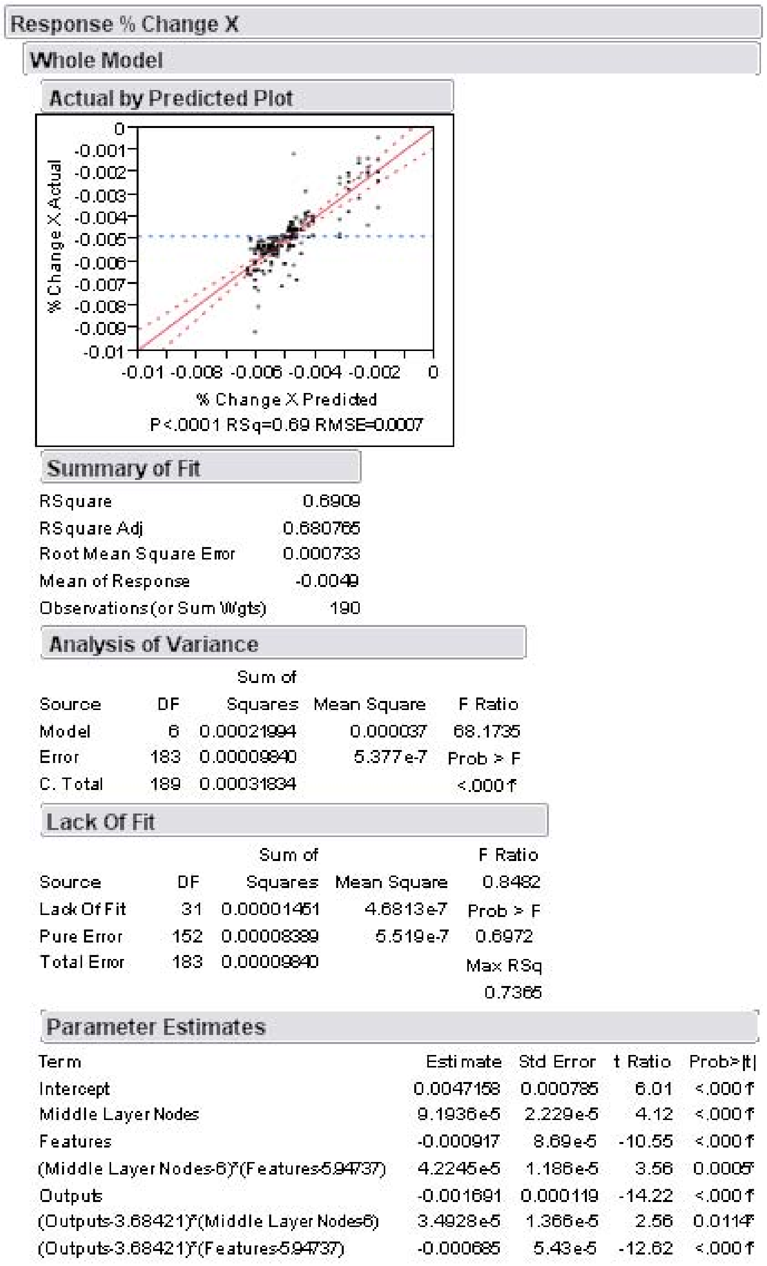
Regression Network Setups

Data Set	Features	Data Points (Exemplars)	Lower Bound	Upper Bound	Outputs
Finance	7	25	3	12	3
XOR	2	150	3	7	2
Hot Dog	5	54	2	10	3
Body Fat	13	252	2	6	2
Owl Problem	4	179	2	10	7

For each data set, different sized middle layers are used. The lower bound and upper bound columns correspond to the different sizes. For example, the Finance Data set is trained separately with middle layers including: 3, 4, 5, 6… 12 middle layer neurons. Five replicates of each setting are run. Also, all network structure with only 1 middle neuron and some with 2 middle neurons were removed, because these configurations did not lead to a successful neural network. Successful implies that the validation accuracy be higher than a random guess. Each network is trained to a point where overtraining is occurring. For a given network, the validation accuracy of the network is recorded after each training epoch, so the max validation accuracy can be recorded.

% CHNG Regression Results

When conducting the regression on the %CHNG value, a Box Cox transformation is utilized due to concerns over the residuals’ non-constant variance in the initial regression model.



Response % Change X

Whole Model

P<.0001 RSq=0.69 RMSE=0.0007

Summary of Fit

RSquare	0.6909
RSquare Adj	0.680765
Root Mean Square Error	0.000733
Mean of Response	-0.0049
Observations (or Sum Wgts)	190

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	6	0.00021994	0.000037	68.1735
Error	183	0.00009840	5.377e-7	Prob > F
C. Total	189	0.00031834		<.0001*

Lack Of Fit

Source	DF	Sum of Squares	Mean Square	F Ratio
				0.8482
Lack Of Fit	31	0.00001451	4.6813e-7	Prob > F
Pure Error	152	0.00008389	5.519e-7	0.6972
Total Error	183	0.00009840		Max RSq
				0.7365

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	0.0047158	0.000785	6.01	<.0001*
Middle Layer Nodes	9.1936e-5	2.229e-5	4.12	<.0001*
Features	-0.000917	8.69e-5	-10.55	<.0001*
(Middle Layer Nodes-6)*(Features-5.94737)	4.2245e-5	1.186e-5	3.56	0.0005*
Outputs	-0.001691	0.000119	-14.22	<.0001*
(Outputs-3.68421)*(Middle Layer Nodes-6)	3.4928e-5	1.366e-5	2.56	0.0114*
(Outputs-3.68421)*(Features-5.94737)	-0.000685	5.43e-5	-12.62	<.0001*

CHNG Regression Model

The output includes the ANOVA, R-Square values, and parameter estimates for the model. The Adj. R-Square of .6808 seems promising. There is a large amount of the variance in the %CHNG variable termination value accounted for by the model. However, the model is hard to interpret because the coefficients are very small. It appears that the number of features and outputs decrease the %CHNG value to stop training while the number of middle nodes slightly increases it. Interestingly, the number of data points did not appear significant in determining the number of training epochs. When trying to apply the results of this model to data used in this research, inverse prediction intervals on %CHNG include zero. Also, the estimated value for %CHNG appears too conservative when using it in practice. This essentially deems the results of this model ineffective. The goal in conducting a regression is to provide an accurate estimate of when to stop training, but this falls short in that regard.

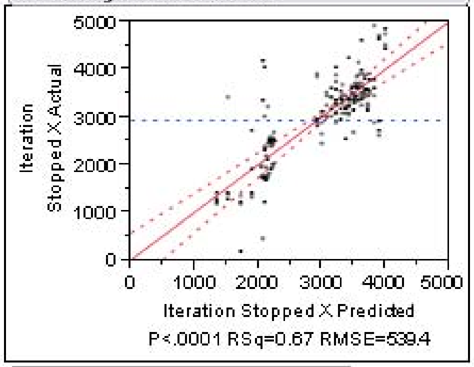
Training Epoch Stopped Regression Results

When conducting the regression on the %CHNG value, a Box Cox transformation is utilized due to concerns over the residuals' non-constant variance in the initial regression model.

Response Iteration Stopped X

Whole Model

Actual by Predicted Plot



Summary of Fit

RSquare	0.674087
RSquare Adj	0.661552
Root Mean Square Error	539.3964
Mean of Response	2925.234
Observations (or Sum Wgts)	190

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	7	109521914	15645988	53.7758
Error	182	52952622	290948.47	Prob > F
C. Total	189	162474536		<.0001*

Lack Of Fit

Source	DF	Sum of Squares	Mean Square	F Ratio
				1.8092
Lack Of Fit	30	13933296	464443	Prob > F
Pure Error	152	39019326	256706	0.0110*
Total Error	182	52952622		Max RSq
				0.7598

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	1314.5217	240.9382	5.46	<.0001*
Middle Layer Nodes	-63.59202	16.40196	-3.88	0.0001*
Features	303.12428	35.69043	8.49	<.0001*
Data Points	11.013463	1.037661	10.61	<.0001*
(Middle Layer Nodes-6)*(Features-5.94737)	-23.39464	8.782949	-2.66	0.0084*
(Features-5.94737)*(Data Points-114.658)	-6.131709	0.451427	-13.58	<.0001*
Outputs	-181.094	33.95468	-5.33	<.0001*
(Outputs-3.68421)*(Middle Layer Nodes-6)	-24.14372	10.11867	-2.39	0.0181*

Training epoch Stopped Regression Model

The output includes the ANOVA, R-Square values, and parameter estimates for the model. The Adj. R-Square of .6741 shows there is a large amount of the variance in the Training epoch Stopped variable accounted for by the model. It appears that the number of middle and output nodes decrease training epochs, while the number of features and data points increases training epochs. Interestingly, the number of data points did not appear significant in determining the number of training epochs. When trying to apply the results of this model to data used in this research, the resulting values appear too conservative when used in practice. Unfortunately, this regression is not as successful as initially hoped.

Appendix B. Blue Dart

Artificial neural networks are a statistical tool patterned after the way a human central nervous system communicates. They have the ability to classify and derive meaning from data without any previous knowledge of the context or origin of it. This is important, because an increasingly technical world contains an abundance of separable data. Neural networks are robust in their application and can discover patterns or trends that are unnoticeable to a human observer or other computer techniques. These powerful tools can be used to do anything from classify pixels in an image to separate or better understand personnel data.

This research provides new techniques to use when implementing neural networks to classify data. The size of a neural network and the amount of time they are trained for can affect results. The techniques presented in this work provide an efficient, structured process for obtaining effective results. For data sets tested thus far, these methods yield efficient neural network structure in minimal training time. Direct application of the techniques proposed in this paper can result in better understanding of information in less time.

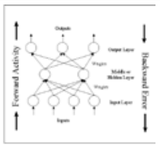
Applications of neural networks include terrain classification in aircraft, anomaly detection in image processing, and target tracking. In all of these realms, decreased training time of the neural networks provides results in minimal time. As computational power increases, the use of neural networks for real time analysis will be a powerful asset in data analysis.

Appendix C. Story Board

Using Upper Layer Weights to Efficiently Construct and Train Feed-forward Neural Networks Executing Backpropagation

2d Lt Harmon Gage
Department of Operational Sciences (ENS)
ADVISOR
Dr. Kenneth Bauer
READER
Dr. J.O. Miller

Research Objectives:
- Determine efficient network structure
- Determine an ideal training termination point
- Combine Methods to create an iterative process to construct neural networks

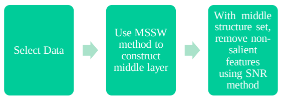


Feedforward Neural Network:

"An important statistical tool for classification. They can adjust themselves to data without any prior knowledge of the input data. Can approximate any function with arbitrary accuracy. "

(Zhang G. P., 2000)

Combined Method


Select Data
Use MSSW method to construct middle layer
With middle structure set, remove non-salient features using SNR method

The combined method integrates the MSSW method, Threshold Stopping Criteria, and Bauer, Alsing, & Greene's SNR feature selection method. The result is an efficient network structure with successful performance.

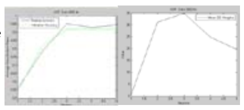
Mean Sum of Squared Weights Method (MSSW)

For a middle layer structure containing *I* middle layer nodes and J upper layer nodes, the MSSW can be calculated as:

$$MSSW = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}(w_{i,j})^2}{I}$$

MSSW Method Summary: Determine number of middle layer nodes where max MSSW method occurs

Max MSSW value coincides with efficient network structure



Generalized Ensemble Method

Stochastic nature of backpropagation algorithm yields varying results across multiple runs. Generalized Ensemble Method fuses multiple runs to create best possible classifier.

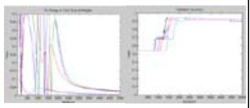
	Combined Method Result		GEM
Data Set	**Avg. Validation Accuracy (5 Runs)**	**Std Dev.**	**Validation Accuracy**
XOR	0.9027	0.0808	0.9189
Body Fat	0.8175	0.0112	0.8254
Finance	0.8833	0.1125	1
Hot Dog	0.6846	0.0568	.6923
Iris	0.9514	0.0279	0.973

Threshold Stopping Criteria

For a training epoch k, the percent change in the sum of upper layer weights is observed:

$$\%CHNG_k = 100 * \left| \frac{\sum_i\sum_j (w_j^i)_k - \sum_i\sum_j (w_j^i)_{k-1}}{\sum_i\sum_j (w_j^i)_{k-1}} \right|$$

Threshold Stopping Criteria:
Train for minimum of:
1. 1000 Epochs
2. Epoch where %CHNG is less than .025% for 50 consecutive training epochs



Q: Why use %CHNG observation?
A: As %CHNG settles, validation accuracy also settles

Conclusion

Data Set	MSSW	Threshold	Combined	GEM
XOR	🟩	🟩	🟩	🟩
Body Fat	🟩	🟩	🟩	🟩
Finance	🟥	🟩	🟨	🟩
Hot Dog	🟩	🟩	🟨	🟩
Iris	🟩	🟨	🟩	🟩

🟩 Success
🟨 Mild Success
🟥 No Success

Success- Efficient or Pareto-Optimal
Mild Success- Realistic trade off in structure size/epochs vs. performance
No Success- Unrealistic trade off in structure size/epochs vs. performance
Results are compared to existing heuristics

Bibliography

Bacauskine, M., & Verikas, A. (2004). Selecting salient features for classification based on neural network committees. *Pattern Recognition Letters , 25*, 1879-1891.

Bauer, K. W., Alsing, S. G., & Greene, K. A. (2000). Feature screening using signal-to-noise ratios . *Neurocomputing* , 29-44.

Baum, E., & Haussler, D. (1989). What sized net gives valid generalization? *Neural Computation , 1* (1), 151-160.

Bebis, G., & Georgiopoulos, M. (1994). Feed-forward neural networks: Why network size is so important. *IEEE Potentials* , 27-31.

Belue, L., & Bauer, K. (1995). Methods of determining input features for multilayer perceptrons. *Neurocomputing , 7* (2), 111-121.

Belue, L., Steppe, J., & Bauer, K. (April 1996). Multivariate statistical techniques for determining neural network architecture and data requirements. *AISB96 Workshop Tutorial Programme.* Brighton, U.K.

Cancelliere, R. (2003). Data Processing and Feature Screening in Function Approximation: An Application to Neural Networks. *Computer & Mathematics with applications , 46*, 455-461.

Caudill, M., & Butler, C. (1992). *Understanding neural networks: Computer explorations.* Cambridge: MIT Press.

Cover, T. (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans. Elec. Comp. , EC-14*, 326-334.

Cybenko, G. (1988). *Continuous valued neural networks with two hidden layers are sufficient.* Tufts University : Department of Computer Science.

Daqi, G., & Shouyi, W. (1998). An optimization method for the topological structures of feed-forward multi-layer neural networks. *Pattern Recognition* , 1337-1342.

Dillon, W., & Goldstein, M. (1984). *Multivariate Analysis: Methods and Applications* . New York: John Wiley & Sons, Inc.

Fahlman, S., & Lebiere, C. (1990). The Cascade-Correlation learning architecture. *Advances in Neural Information Processing Systems II* , 524-532.

Fanguy, R., & Kubat, M. (2002). Modifying Upstart for Use in Multiclass Numerical Domains. *FLAIRS*, (pp. 339-343).

Frean, M. (1990). The Upstart Algorithm: A Method for Constructing and Training. *Neural Computation , 2*, 198-209.

Gao, P., Chen, C., & Qin, S. (2010). An Optimization Method of Hidden Nodes for Neural Network. *2010 Second International Workshop on Education Technology and Computer Science* (pp. 53-56). Hubei, China: IEEE.

Gorman, P. R., & Sejnowski, T. (1988). Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets. *Neural Networks* , 75-89.

Guler, I., & Ubeyli, E. D. (2005). Feature saliency using signal-to-noise ratios in automated diagnostic systems developed for ECG beats . *Expert Systems with Applications* , 295-304.

Huang, G.-B., & Babri, H. A. (1998). Upper Bounds on the Number of Hidden Neurons in Feedforward Networks with Arbitrary Bounded Nonlinear Activation Functions. *Transactions on Neural Networks* , 224-229.

Ileana, I., Rotar, C., & Incze, A. (2004). The Optimization of Feed Forward Neural Networks Structure Using Genetic Algorithms. *International Conference on Theory and Applications of Mathematics and Informatics*, (pp. 223-234). Thessaloniki, Greece.

Kavzoglu, T. (1999). Determining Optimum Structure for Artificial Neural Networks. *25th Annual Technical Conference and Exhibition of the Remote Sensing Society.* Cardiff, UK.

Kocur, C., Roger, S., Myers, L., Burns, T., Hoffmeister, J., Bauer, K., et al. (1996). Using neural networks to select wavelet features for breast cancer diagnosis. *IEEE Trans. Neural Networks , 15* (3), 95-102.

Kurkova, V. (1992). Kolmogorov's thorem and multilayer neural networks. *Neural Networks , 5* (3), 501-506.

Kurkova, V. (1992). Komogorov's thorem and multilayer neural networks. *Neural Networks , 5* (3), 501-506.

Law, A. M. (2007). *Simulation Modeling & Anlaysis* (4th ed.). New York, NY: McGraw Hill.

Lippmann, R. P. (1987). An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine , 3* (4), 4-22.

Mahajan, P. S., & Ingalls, R. G. (2004). Evaluation Of Methods Used To Detect Warm-Up Period In Steady State Simulation. *Winter Simulation Conference* (pp. 663-671). Stillwater: OK.

Mirchandani, G., & Cao, W. (1989). On Hidden Nodes for Neural Nets. *IEEE Transactions on Circuits and Systems* , 661-664.

Parekh, R., Yang, J., & Honavar, V. (2000). Constructive Neural-Network Learning Algorithms for Pattern Classification. *IEEE Transactions on Neural Networks , 11* (2), 436-451.

Perrone, M. P., & Cooper, L. N. (1992). *When Networks Disagree: Ensemble Methods for Hybrid Neural Networks.* Providence, RI : Chapman-Hall.

Prechelt, L. (1998). Early Stopping - But When? In G. Orr, & R. Muller, *Neurlan Networks: Tricks of the Trade* (pp. 55-69). Springer-Verlag Berlin Heidelberg.

Reed, R. (1993). Pruning Algorithms - A Survey. *IEEE Transactions on Neural Networks* , 740-746.

Sartori, M. A., & Antsaklis, P. J. (1991). A Simple Method to Derive Bounds on the Size and to train Multilayer Neural Networks. *IEEE Transactions on Neural Networks* , 467-471.

Setiono, R., & Liu, H. (1997). Neural-Network Feature Selector. *IEEE Transactions on Neural Networks , 8* (3), 654-662.

Sietsma, J., & Dow, R. J. (1991). Creating Artificial Neural Networks That Generalize. *Neural Networks , 4*, 67-79.

Sontag, E. (1992). Feedforward nets for interpolation and classification. *J. Comp. Syst. Sci. , 45*, 20-48.

Sprinkhuizen-Kuyper, I. G., & Boers, E. J. (1999). A Local Minimum for the 2-3-1 XOR network. *IEEE TRANSACTIONS ON NEURAL NETWORKS , 10* (4), 968-971.

Steppe, J. M. (1994). *Feature and Model Selection in Feedforward Neural Networks.* Wright-Patterson AFB, OH: AFIT.

Steppe, J. M., Bauer, K. W., & Rogers, S. K. (1996). Integrated Feature and Architecture Selection. *IEEE Transacations on Neural Networks* , 1007-1014.

Steppe, J., & Bauer, K. (1996). Improved feature screening in feedforward neural networks. *Neurocomputing , 13*, 47-58.

Svozil, D., Kvasnicka, V., & Jiri, P. (1997). Tutorial Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems* , 43-62.

Tarr, G. (1991). Multilayered feeforward neural networks for image sementation. *Ph.D. Dissertation* .

Teoh, E. J., C, T. K., & Xiang, C. (2006). Estimating the Number of Hidden Neurons in a Feedforward Network Using the Singular Value Decomposition. *IEEE Transactions on Neural Networks* , 1623-1629.

Ubeyli, E. D. (2008). Measuring saliency of features extracted by model-based methods from internal carotid arterial Doppler signals using signal-to-noise ratios. *Digital Signal Processing , 18*, 2-14.

Ubeyli, E. D. (2008). Measuring Saliency of Features Using Signal-to-noise Ratios for Detection of Electrocardiographic Changes in Partial Epileptic Patients. *Journal of Medical Systems , 32* (6), 463-470.

Vapnik, V., & Chervonenkis, A. (1971). On the convergence of relative frequencies of events to their probabilities. *Theory Probab. Its Appl. , 16* (2), 264-280.

Verikas, A., & Bacauskiene, M. (2002). Feature selection with neural networks. *Pattern Recognition Letters , 23*, 1323-1335.

Welch, P. (1981). *On the Problem of the Initial Transient in Steady-State Simulation.* Yorktown Heights, New York: IBM Watson Research Center.

White, H. (1993). *Artificial Neural Networks Approximation & Learning Theory.* Cambridge, USA: Blackwell Publishers.

Yang, J., & Honavar, V. (1991). *Experiments with the Cascade-Correlation Algorithm.* Ames, IA: Iowa State University Department of Computer Science.

Zhang, G. P. (2000). Neural Networks for Classification: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics* , 451-462.

Zhang, L., Jiang, J.-H., Liu, P., Liang, Y.-Z., & Yu, R.-Q. (1997). Multivariate nonlinear modelling of flourescence data by neural network with hidden node pruning algorithm. *Analytica Chimica Acta* , 29-39.

Vita

Second Lieutenant Harmon John Ambrose Gage graduated from Bob Jones High School in Madison, Alabama in 2005. He entered undergraduate studies at the United States Air Force Academy in Colorado Springs, Colorado where he graduated with a Bachelor of Science degree in Operation Research in May 2009 and received his commission as an Air Force Officer. In August 2009, he entered the Graduate School of Engineering and Management, Air Force Institute of Technology. Upon graduation, he will be assigned to the 36th Electronic Warfare Squadron at Eglin AFB, FL.

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)* 24-03-2011	2. REPORT TYPE **Graduate Research Project**	3. DATES COVERED *(From – To)* Sep 2009 – Mar 2011
4. TITLE AND SUBTITLE USING UPPER LAYER WEIGHTS TO EFFICIENTLY CONSTRUCT AND TRAIN FEEDFORWARD NEURAL NETWORKS EXECUTING BACKPROPAGATION		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Gage, Harmon, J.A., 2d Lt, USAF		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/EN), 2950 Hobson Street, Building 642, WPAFB OH 45433-7765		8. PERFORMING ORGANIZATION REPORT NUMBER AFIT-OR-MS-ENS-11-06
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) NASIC/DAIA, Attn: John Jacobson, 4180 Watson Way, Wright-Patterson AFB OH 4533-5648		10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Feed-forward neural networks executing back propagation are a common tool for regression and pattern recognition problems. These types of neural networks can adjust themselves to data without any prior knowledge of the input data. Feed-forward neural networks with a hidden layer can approximate any function with arbitrary accuracy.

In this research, the upper layer weights of the neural network structure are used to determine an effective middle layer structure and when to terminate training. By combining these two techniques with signal-to-noise ratio feature selection, a process is created to construct an efficient neural network structure. The results of this research show that for data sets tested thus far, these methods yield efficient neural network structure in minimal training time. Data sets used include an XOR data set, Fisher's Iris problem, a financial industry data set, among others.

15. SUBJECT TERMS

Neural Network, Hidden Layer, Feed-Forward, Backpropagation, Training Epochs, SNR feature selection

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Kenneth W. Bauer (ENS)
U	U	U	UU	102	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636, ext 4631; e-mail: Kenneth.Bauer@afit.edu